(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 204 210 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
08.05.2002 Bulletin 2002/19

(51) Int Cl.[7]: **H03M 13/27**

(21) Application number: **01307697.1**

(22) Date of filing: **11.09.2001**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE TR
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **18.09.2000 US 233369 P**
18.07.2001 US 908003

(71) Applicant: **LUCENT TECHNOLOGIES INC.**
Murray Hill, New Jersey 07974-0636 (US)

(72) Inventors:
- **Nicol, Christopher J.**
Springwood, NSW 2777 (AU)
- **Bickerstaff, Mark Andrew**
Carlingford, NSW 2118 (AU)
- **Xu, Bing**
Tempe, AZ 85283 (US)
- **Yan, Rang-Hong**
Holmdel, NJ 07733 (US)

(74) Representative:
Watts, Christopher Malcolm Kelway, Dr. et al
Lucent Technologies (UK) Ltd,
5 Mornington Road
Woodford Green Essex, IG8 0TU (GB)

(54) Architecture for a communications device

(57) The present invention discloses a single unified decoder for performing both convolutional decoding and turbo decoding in the one architecture. The unified decoder can be partitioned dynamically to perform required decoding operations on varying numbers of data streams at different throughput rates. It also supports simultaneous decoding of voice (convolutional decoding) and data (turbo decoding) streams. This invention forms the basis of a decoder that can decode all of the standards for TDMA, IS-95, GSM, GPRS, EDGE, UMTS, and CDMA2000. Processors are stacked together and interconnected so that they can perform separately as separate decoders or in harmony as a single high speed decoder. The unified decoder architecture can support multiple data streams and multiple voice streams simultaneously. Furthermore, the decoder can be dynamically partitioned as required to decode voice streams for different standards.

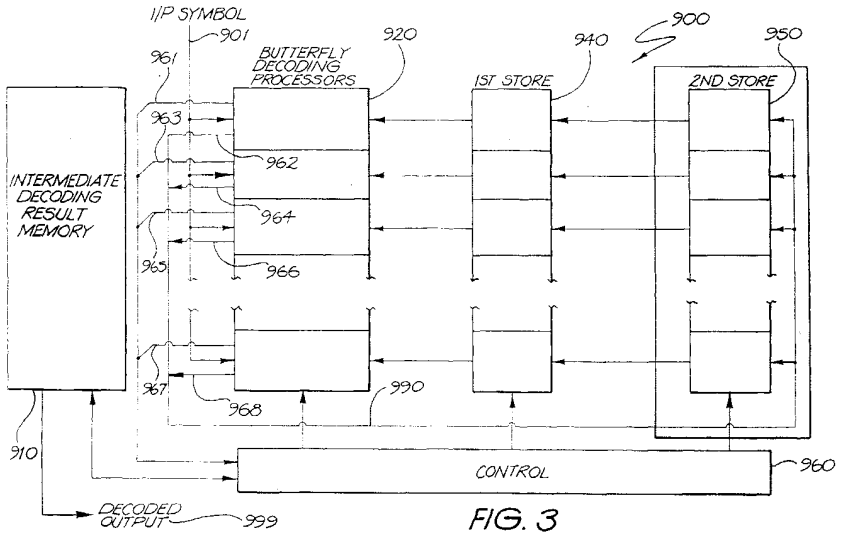


FIG. 3

EP 1 204 210 A1

Description

Technical Field of the Invention

[0001] The present invention relates generally to wireless communications and, in particular, to a decoder architecture for wireless communications systems.

Background Art

[0002] Communication systems deal with the transmission of information from a transmitter to a receiver. The transmission medium through which the information passes often contains many sources of noise, including cosmic radiation, Additive White Gaussian Noise (AWGN), Rayleigh scattering (multipath propagation) and electromagnetic noise. The presence of these noise sources corrupts or prevents the transmission of the desired information, thus limiting the ability to communicate.

[0003] It is well known in the art that coding of the information to be transmitted, through the addition of redundant information calculated from the source information, improves the ability to successfully receive the transmitted information. Decoding uses the redundant information to detect the presence of errors or estimate the most probable emitted bits, given those received. Errors are detected when the transmitted redundancy is different from that subsequently calculated with the received data.

[0004] The weight of a codeword is a measure of the capacity to recover data from the codeword. A codeword with a high number of bits has a high weight. A low weight codeword exhibits a low ability to recover data, whereas, conversely, a high weight codeword exhibits improved recovery of data.

[0005] Automatic-repeat-request (ARQ) coding schemes employ an error-detection code. If the presence of an error is detected in the information received, a message requesting retransmission of the relevant information is sent from the receiver to the transmitter. ARQ coding schemes are relatively simple, but require the use of a feedback channel and deliver variable and comparatively slow throughput.

[0006] Forward error correction (FEC) coding schemes are used to encode information in systems in which propagation delays and latency are of concern. The receiver is able to detect and correct errors, without requiring a feedback channel.

[0007] Coding schemes can be broadly categorised into block codes and convolutional codes.

[0008] Block codes map a message of k information bits into a structured sequence of n bits, where n>k. The code is referred to as a (n,k) code. The ratio (n-k)/k is called the redundancy of the code and the ratio of information bits to the total number of bits. k/n, is called the code rate. The extra bits inserted provide redundancy and are used by the decoder to provide error detection and correction. The redundant bits added during encoding are only dependent on the k information bits in the message block. Block codes are often used to detect errors when ARQ is implemented.

[0009] Convolutional encoding generates a block of n code bits in a given period of time from k information bits, where n and k are typically small. The block of n bits generated by the encoder is dependent not only on the k information bits of the time period, but also on the message blocks generated during a predefined number of preceding time periods. The memory thus imparted on the coding enables errors to be corrected based on allowable sequences of codes. Convolutional decoding may be performed using either a Viterbi algorithm or LogMAP algorithm.

[0010] Convolutional codes are preferred for wireless voice communications systems in which the retransmission of data and its associated delay is intolerable. Block codes are capable of delivering higher throughput and are preferred for the transmission of data where latency is less of a concern.

[0011] Turbo codes, also known as parallel concatenated codes, are a class of codes whose performance is very close to the Shannon capacity limit. Turbo coders are implemented by connecting convolutional encoders either in parallel or series to produce concatenated outputs. Bit sequences passing from one encoder to another are permuted by an interleaver. In this manner, low-weight code words produced by a single encoder are transformed into high-weight code words. Turbo decoding thus takes two low weight codewords and obtains the effect of a much higher weight codeword.

[0012] At present, consumer wireless communication systems are primarily concerned with the transmission of voice. Such wireless communication systems include Advanced Mobile Phone Service (AMPS), Global System for Mobile Communication (GSM) and Code Division Multiple Access (CDMA). These represent the first (1G) and second (2G) generation systems. With the convergence of data and voice communication systems, the second-and-a-half generation (2.5G) and third generation (3G) systems are emerging in which the transmission of data is becoming a more important concern. In order to achieve superior error performance at higher transmission rates, turbo block encoding is preferred. The latency endemic to block coding is not as significant an issue as it is with the transmission of voice. New, third generation mobile wireless standards, like Universal Mobile Telecommunication Service (UMTS) and CDMA2000 require turbo encoding for data streams and convolutional encoding for voice streams. These systems

require a complex turbo decoder for data and a Viterbi decoder for voice. Furthermore. backward compatibility requires that second generation standards are also supported.

[0013] The transmission of voice and data provides conflicting requirements of transmission rate versus latency and propagation delay. The current mode of addressing these problems is to provide separate encoding systems: turbo encoding for data streams and convolutional encoding for voice streams. Consequently, different decoders are also required, resulting in a multiplicity of hardware platforms and thus increased costs for telecommunications operators.

Summary of the Invention

[0014] The prior art's problem with decoding is overcome, in accordance with the principles of the invention, by a single unified decoder for performing both convolutional decoding and turbo decoding in the one architecture. The unified decoder architecture can support multiple data streams and multiple voice streams simultaneously. The unified decoder can be partitioned dynamically to perform required decoding operations on varying numbers of data streams at different throughput rates. The unified decoder also supports simultaneous decoding of voice (convolutional decoding) and data (turbo decoding) streams. Advantageously, the unified decoder can be used to decode all of the standards for TDMA, IS-95, GSM, GPRS, EDGE, UMTS, and CDMA2000. The preferred embodiment is modular and thus readily scalable.

[0015] The reconfigurable architecture is capable of decoding data communication signals transmitted according to one of the plurality of coding schemes. The architecture includes: a trellis processing arrangement for receiving an input signal derived from the transmitted signals and new path metrics for determining intermediate decoded results using path metrics; an intermediate store for receiving modified decoded results and for providing a decoding output; and a controller. The controller is coupled to the trellis processing arrangement and is able to configure the architecture to perform one of convolutional or turbo decoding by forming the new path metrics using generated path metrics output from the trellis processing arrangement, determining the modified decoded results from the intermediate decoded results, and determining the decoded output from a selected one of the modified decoded results.

[0016] In accordance with one embodiment, a telecommunications decoding device consists of decoding processors and at least one store arranged in a processing loop. The device includes a control arrangement capable of reconfiguring the processing loop such that the processing loop can operate in accordance with at least two different coding schemes.

[0017] Another embodiment is directed to a telecommunications decoding device, which is capable of being scaled in either one or both of the time and space domains. In accordance with the principles of the invention, the decoding device consists of atomic processing units, each of which is capable of decoding input data provided according to one of a plurality of coding schemes. The atomic processing units may be stacked together and interconnected using an hierarchical switching structure. The individual processing units can perform independently as separate decoders. Alternatively, the individual processing units may be combined to form a signal high speed decoder with a predetermined processor being dominant. The flexibility of the architecture allows multiple parallel streams of input symbols to be processed contemporaneously or a single stream of input symbols to be processed more quickly by combining a plurality of processing units.

[0018] Advantageously, embodiments can perform both Viterbi and Log Map calculations by enabling one of two processors to evaluate a trellis in accordance with a determined coding arrangement of presented input symbols. The flexibility afforded by the invention allows telecommunications operators to reduce hardware costs and respond dynamically to variations in the coding of transmitted signals.

Brief Description of the Drawings

[0019] A number of preferred embodiments of the present invention will now be described with reference to the drawings, in which:

Fig. 1 is a schematic block diagram representation of a communication network employing multiple protocols;
Fig. 2A is a schematic block diagram representation of a communication system employing coding;
Fig. 2B is a schematic block diagram representation of a generic Viterbi decoder in a communication system employing coding;
Fig. 2C is a schematic block diagram representation of a generic turbo decoder in a communication system employing coding;
Fig. 3 is a schematic block diagram representation of a unified decoder;
Fig. 4 is a schematic block diagram representation of an architecture for a unified decoder;
Fig. 5A is a schematic block diagram representation of a butterfly processor of Fig. 4;
Fig. 5B is a schematic block diagram representation of an Add-Compare-Select (ACS) unit of Fig. 4;

Fig. 6A is a representation of a 32-state trellis and its corresponding butterfly processors and path metrics;
Fig. 6B shows the resultant path metric locations;
Figs 7A-7E are representations of the in-place path metric addressing at times t=1 to t=5 respectively;
Fig. 7F is a representation of the addressing of the path metric columns;
Figs 8A-8F are representations of the in-place path metric addressing of a reverse trellis configuration;
Fig. 9A is a schematic block diagram representation of an Intermediate Decoding Memory Processor of Fig. 4;
Fig. 9B is a schematic block diagram representation of an exploded view of Fig. 9A showing a Window Memory Subsystem, Traceback Controller and Interleaver Controller;
Fig. 9C is a schematic block diagram representation of a Traceback Controller of Fig. 9B;
Fig. 9D is a schematic block diagram representation of an Interleaver of Fig. 9B;
Fig. 9E is an exploded view of an Interleaver address controller of Fig. 9D;
Fig. 9F is a schematic block diagram representation of a Window Memory Subsystem of Fig. 9B;
Fig. 10A is a schematic block diagram representation of a LogLikelihood processor of Fig. 4 for a single row decoder;
Fig. 10B is a schematic block diagram representation of an Add-Compare-Select Node unit of Fig. 10A;
Fig. 10C is a schematic block diagram representation of a LogLikelihood processor of Fig. 4 for an eight row decoder;
Fig. 10D is a schematic block diagram representation of an ACS unit of Fig. 10A;
Fig. 11 is a schematic block diagram representation of a bank of butterfly decoding processors of Fig. 4;
Fig. 12 is a schematic block diagram representation of a Reverse Address Processor of Fig. 4;
Fig. 13 is a schematic block diagram representation of Normalisation Subtractors of Fig. 4;
Fig. 14 is a schematic block diagram representation of a Comparator of Fig. 4;
Fig. 15 is a schematic block diagram representation of a Path Metric Memory of Fig. 4;
Fig. 16 is a schematic block diagram representation of a Forward Address Processor;
Fig. 17 is a schematic block diagram representation of a Comparator (ACS level) of Fig. 4;
Fig. 18 is a schematic block diagram representation of an Input Symbol History of Fig. 4;
Fig. 19 is a schematic block diagram representation of a LogLikelihood Ratio Processor of Fig. 4;
Figs. 20A and 20B illustrate use of multiple decoders to implement a single Turbo decoder;
Fig. 21 is a schematic block diagram representation of two interconnected decoders operating together as a single decoder (16-state trellises every cycle);
Fig. 22 is a schematic block diagram representation of four interconnected decoders operating together as a single decoder for even higher performance decoding (32-state trellises every cycle); and
Figs 23A and 23B are schematic block diagram representations of a non-systematic encoder.

Detailed Description

[0020] The preferred embodiment provides a unified decoder architecture for wireless communication systems. The unified decoder implements the decoding required for convolutional encoded and turbo encoded data streams. The unified decoder architecture can support multiple data streams and multiple voice streams simultaneously. Furthermore, the decoder can be dynamically partitioned, as required, to decode voice streams for different standards. The preferred embodiment is modular and thus readily scalable.

[0021] Fig. 1 shows a wireless communication network 100. A UMTS base station 110 contains a transmitter/receiver 112, which contains a decoder module 150a. The transmitter/receiver 112 communicates via a switching network 160 with another UMTS transmitter/receiver 146 located in a remote base station 140 and containing a decoder module 150f. The transmitter/receiver 112 also communicates with a mobile handset 160a, which contains a decoder module 150i. The transmitter/receiver 146 communicates with another mobile handset 160f, which contains a decoder unit 150m.

[0022] The base station 140 contains further transmitter/receivers 142 and 144, containing decoder units 150d and 150e respectively. Transmitter/receiver 142 is configured to operate as a CDMA transmitter/receiver and communicates via the switching network 160 with remote CDMA base station 130 containing CDMA transmitter/receiver 132 and decoder unit 150c. The transmitter/receiver 142 also communicates with a mobile handset 160d, containing decoder unit 150j. The transmitter/receiver 132 communicates with a mobile handset 160c, containing decoder unit 150g.

[0023] Transmitter/receiver 144 communicates via the switching network 160 with remotely located base station 120, containing transmitter/receiver 122 and decoder unit 150b. The transmitter/receiver 144 also communicates with a mobile handset 160e. containing a decoder unit 150k. The transmitter/receiver 122 communicates with a mobile handset 160b, containing decoder unit 150h.

[0024] The decoder units 150a, 150b, 150c, 150d, 150e, 150f, 150g, 150h. 150i, 150j, 150k and 150m located in the transmitter/receivers 112, 122, 132, 142, 144 and 146 and mobile handsets 160a...160f are embodiments of the unified

decoder architecture, which have been configured to conform to different cellular network standards.

[0025] The unified decoder architecture of Fig. 1 offers telecommunication companies operating multiple network standards great benefits in flexibility and cost reduction as the same decoder block can be used to implement many different coding schemes in different network components.

[0026] Fig. 2A shows a typical communication system 200 in which coding is used to improve the transmission of information from a transmitter 210 to a receiver 270. The transmitter 210 has an information source 205 supplying an input data stream to an encoder 220, in which redundant information is added to the input data stream in accordance with a predefined coding algorithm so as to improve the ability to detect and correct errors which may occur during the transmission of information as a result of noise sources present in a communication channel 240. The encoded input stream is then modulated 230 to impress the encoded data stream onto a waveform to be transmitted. The encoded information is transmitted over a channel 240, which has many sources of noise 280 acting upon it. The channel 240 couples to a receiver 270 having a demodulator 250 complementing the modulator 230, the demodulator 250 producing an output to a decoder 260, which outputs a received information signal 275.

[0027] Fig. 2B shows the communication system 200, in which the decoder 260 is a generic Viterbi decoder. The input to the Viterbi decoder 260 is coded information received from the channel 240. The Viterbi decoder includes a branch metric calculator (BMC) unit 289, whose output is presented to an add-compare-select (ACS) unit 291. A state controller 290 provides inputs to the BMC unit 289, the ACS unit 291 and a path metric memory 292. The path metric memory 292 acts as a double buffer and interchanges information with the ACS unit 291. A borrow output 294 of the ACS unit 291 is presented to a traceback memory and controller 293, whose output is the received information signal 275.

[0028] Fig. 2C shows a Turbo decoding configuration of the decoder 260 of Fig. 2A. A received symbol in a turbo decoder consists of systematic data, representing the actual data being transmitted, and parity data, which represents the coded form of the data being transmitted. A first input 261, being the parity data of the received symbol, is presented to a demultiplexer 263. A first output 264 of the demultiplexer 263 is presented to a first decoder 266. A second input 262, being the systematic data of the received symbol, is presented to the first decoder 266. A recursive input 277 is also presented to the first decoder 266. The output 267 of the first decoder 266 is then presented to an interleaver 268, whose output 269 is presented to a second decoder 271. A second output 265 of the demultiplexer 263 is also presented to the second decoder 271. A first output 272 of the second decoder 271 is presented to a first deinterleaver 274, whose output is the recursive input 277. A second output 273 of the second decoder 271 is presented to a second deinterleaver 276. The output of the second deinterleaver 276 is presented to a slicer 278, which applies a threshold to a soft output to convert it to a hard output, being a received information signal 275.

[0029] The unified decoder architecture of the preferred embodiment is intended to replace the decoder 260 in wireless communication systems having both voice and data capabilities and exploits the similarity in the computations needed for Viterbi decoding and LOG-MAP Turbo decoding so that memory and processing units are used efficiently when configured for either of such schemes. LogMAP is an algorithm which may be utilised in the decoding of convolutional codes. LogMAP is also used in one half cycle of a turbo decode iteration. Processors within the preferred embodiment are stacked together and interconnected using a hierarchical switching structure so that they can perform independently as separate decoders or, alternatively, they may be combined to form a single high speed decoder, with a predetermined processor being dominant.

[0030] Fig. 3 shows the block architecture of a unified decoder structure 900 in accordance with an embodiment of the present invention. A multi-bit input symbol 901 from an alphabet of a prior agreed coding scheme for a particular transmission is broadcast to a bank of butterfly decoding processors 920. The bank of butterfly decoding processors 920 also receives as inputs the outputs of a bank of first stores 940. A control unit 960 provides inputs to each of an intermediate decoding result memory 910, the bank of butterfly decoding processors 920, the bank of first stores 940 and a bank of a second stores 950. The control unit 960 issues appropriate control signals via the inputs to implement convolutional or turbo coding, as desired.

[0031] The embodiment depicted in Fig. 3 is that of a single row decoder. When multiple decoder rows are interconnected to form a single decoder, each of the decoder rows in the single decoder is presented with the same multi-bit input symbol 901. When multiple decoder rows are acting as multiple decoders, each decoder being implemented is presented with a separate multi-bit input symbol 901.

[0032] The bank of butterfly decoding processors 920 produces first outputs 962, 964, 966 and 968, which are transmitted via a bus 990 to the bank of second stores 950. Outputs of the bank of second stores 950 are presented as inputs to the bank of first stores 940. A generic embodiment of the decoder typically uses a bank of first stores 940 and a bank of second stores 950 in a double buffering mode.

[0033] The bank of butterfly decoding processors 920 produces second outputs 961, 963, 965 and 967, which are intermediate decoding results presented to the control unit 960.

[0034] The bank of butterfly decoding processors 920 and the loop feedback connection via at least one of the stores form a loop functioning as a trellis processor.

[0035] The intermediate decoding result memory 910 produces a decoded output 999. The intermediate decoding result memory 910 may provide recursive results to the control unit 960 when computing a LogMAP algorithm, as described later.

[0036] Fig. 4 shows the block architecture of a unified decoder 1200 in accordance with a preferred embodiment of the present invention. A control unit 1210 of the unified decoder 1200 receives a number of inputs, including rate 1201, constraint length 1202, convolutional or turbo selector 1203, polynomials 1204, trellis direction 1205, number of iterations 1206, block length 1207, clock 1208 and reset 1209. The rate 1201 indicates how much information is used to represent a single data bit present in a transmitted block. The constraint length 1202 indicates how many previous input symbols are used to encode a presented input information bit and is, thus, also an indicator of the complexity of the trellis being processed to decode a given input symbol. The polynomials 1204 are generator polynomial coefficients used in the decoding process. The number of iterations 1206 determines how many loops are executed by the decoder 1200 when operating in turbo mode. A larger value for the number of iterations 2106 indicates a more accurate decoded output 1294 at the cost of increased computational time.

[0037] The control unit 1210 is interconnected to an Intermediate Decoding Memory and Processor 1240, LogLikelihood Processors 1250a and 1250b, a bank of multiplexers 1250c, a Comparator 1247, Butterfly Decoding Processors 1260, a Reverse Address Processor 1270, Normalisation Subtractors 1278, a bank of multiplexers 1278a, a Path Metric Store 1280, a Forward Address Processor 1290, a LogLikelihood Ratio Processor 1297 and an Input Symbol History 1298. The Control unit 1210 is able to reconfigure the architecture of the unified decoder 1200 via these connections to implement either a convolutional decoder or a turbo decoder, as desired.

[0038] Input symbols 1299 are presented to an Input Symbol History 1298, which functions.as a double buffer to ensure that a constant data flow is maintained. The Input Symbol History 1298 also receives an Input Symbol History Bank Select 1211, an Input Symbol History Address 1219, an Input Symbol History Clock 1223 and an Input Symbol History Reset 1225 from the control unit 1210. The Input Symbol History 1298 produces a first output 1291a, which is presented to Butterfly Decoding Processors 1260, and a second output 1291b, which is presented to LogLikelihood Processor 1250a.

[0039] The Butterfly Decoding Processors 1260 also receive as inputs reverse trellis path metrics 1265 from the Reverse Address Processor 1270, and extrinsic information 1242 from the Intermediate Decoding Memory and Processor 1240. The control unit 1210 also provides a number of inputs to the Butterfly Decoding Processors 1260. including a Butterfly Reset 1215, Butterfly Rate 1216, Butterfly Clock 1217, Butterfly Polynomials 1218, Butterfly Constraint 1220, Butterfly Mode 1221 and beta-phase enable 1235.

[0040] The Butterfly Decoding Processors 1260 produce new multiple bit path metrics for a corresponding state in a trellis diagram, the new path metrics being output on the 32 bit buses 1266 and 1267, which are connected to a Comparator 1247 and a bank of multiplexers 1250c. The Butterfly Decoding Processors 1260 also produce decision bits 1255, which are presented as inputs to the Intermediate Decoding Memory and Processor 1240.

[0041] In a first phase of a LogMAP computation, the Butterfly Decoding Processors 1260 compute gammas and alphas. In a second phase, the Butterfly Decoding Processors 1260 calculate betas using dummy betas computed by LogLikelihood Processor 1250a and LogLikelihood Processor 1250b in the first phase.

[0042] Each butterfly processor within the bank of butterfly processor 1260 contains two Add-Compare-Select units (shown as ACS) 320 and an intermediary Branch-Metric Calculator (BMC) 330, as depicted in Fig. 5A. The BMC 330 executes the same functions as the Branch Metric Units (BMUs) in well-known Viterbi decoders and each ACS 320 performs path metric calculation for trellis decoding.

[0043] Fig. 5A shows an exemplary butterfly unit of the butterfly processors 1260 of Fig. 4, having two Add-Compare-Select units 320 and an intermediary Branch-Metric calculator 330. Each of the Add-Compare-Select units 320 is presented with input path metric-0 1265a and input path metric-1 1265b. The Input Symbol 1291a and extrinsic information 1242 are broadcast to each of the Branch Metric Calculators 330 in the bank of butterfly processors 1260. The intermediary Branch-Metric calculator is also presented with a butterfly rate 1216, a butterfly constraint 1220 and butterfly polynomials 1218.

[0044] Each state in a column of a trellis has a pair of branch metrics leading to it. Each of the individual branch metrics has a symbol associated with it. Therefore, when navigating a trellis in a given direction, one of two possible symbols is expected for a state under consideration, depending on the previous states. The BMC 330 determines a measure of the proximity of the received input symbol 1291a to an expected symbol. The BMC 330 generates an output branch metric-0 406, which is presented to a first ACS unit-0 320 and a second ACS unit-1 320 on a bus being *m* bits wide. The BMC 330 exploits the symmetry of the trellis and produces a second branch metric-1 402, by arithmetically inverting the branch metric-0 406. The branch metric-1 402 is presented to the first ACS unit-0 320 and the second ACS unit-1 320 on a bus which is also *m* bits wide. A butterfly mode 1221 is presented to each of the ACS units 320 to configure them appropriately for the coding scheme in use. The ACS units 320 and the BMC unit 330 also receive a butterfly reset 1215, a butterfly clock 1217 and a beta-phase enable 1235.

[0045] Each of the ACS units 320 generates two outputs which, for ACS 0 in Fig. 5A, consist of a first output 1255a

and a second output 1267a. The first output 1255a is a decision bit which is the value of the comparison borrow bit, indicating which of the upper or lower potential path metrics is selected. A decision bit with a value of 0 corresponds to the lower potential path metric being selected, whereas conversely a value of 1 corresponds to the upper potential path metric being selected. The second output 1267a is a new multiple bit path metric for a corresponding state in a trellis diagram. ACS 1 produces corresponding outputs 1255b and 1267b.

[0046] Fig. 5B shows an architecture of an ACS unit-0 320 of Fig. 5A. Two pairs of inputs 402 and 1265b, and 406 and 1265a are presented to respective Adders 410 and 412. The first pair of inputs consists of the branch metric-1 402 and path metric-1 1265b, whereas the second pair of inputs consists of branch metric-0 406 and path metric-0 1265a. The constituent elements of each of the input pairs are added in respective adders 410 and 412, the corresponding outputs 411 and 413 of the adders 410 and 412 being presented to a Full Subtractor 414. The outputs 411 and 413 are also presented to a first two-to-one multiplexer 420. A borrow output 1255a of the Full Subtractor 414 is fed to the first multiplexer 420 to compute a maximum MAX of the input values. The borrow bit 1255a is also presented as an output of the ACS unit 320, with a value of indicating that the lower path metric has been chosen and a value of 1 indicating that the upper path metric has been selected. A second output 415 of the Full Subtractor 414, representing the difference of the two adder results 411 and 413, is presented to a Log-sum correction table 440, which adjusts the result of the new path metric, when the output of the Full Subtractor 414 is small, to produce a more accurate result in the Log domain for LogMAP decoding. An output 441 of the Log-sum correction table 440 is presented to an Adder 460. An output 421 of the first multiplexer 420 is presented to the Adder 460 and to a second two-to-one multiplexer 450. A result 461 from the Adder 460 is then presented as a second input to the second multiplexer 450. A control signal, being butterfly mode 1221, is also presented as an input to the second multiplexer 450 and is used to determine whether the Viterbi or LogMAP coding scheme is being implemented. The second multiplexer 450 forms an output 451, which feeds an Accumulate Register 470 and a further multiplexer 480. The Accumulate Register 470 receives a butterfly reset 1215 and produces an output 472 to the multiplexer 480. The multiplexer 480 receives a beta-phase enable 1235 as a select signal that selects the output 451 when inactive and the output 472 from the Accumulate register 470 when active. The selected output of the multiplexer 480 is the output path metric 1267a of the ACS unit 320.

[0047] The bank of multiplexers 1250c receives a select signal 1258 from the control unit 1210, which is used to select either the butterfly path metrics 1266 and 1267 output from the Butterfly Processors 1260 or the path metrics produced by the LogLikelihood Processor-0 1250a and LogLikelihood Processor-1 1250b. During a Viterbi calculation, the butterfly path metrics 1266 and 1267 are selected. In the first phase of a LogMAP computation, butterfly path metrics 1266 and 1267 are chosen whilst the Butterfly Decoding Processors 1260 compute gammas and alphas. Contemporaneously, LogLikelihood Processor 1250a calculates dummy betas. At the end of the first phase, the path metrics produced by the LogLikelihood Processor-0 1250a are selected by the bank of multiplexers 1250c to be broadcast to enable the calculation of betas in the second phase of the LogMAP computation.

[0048] The bank of multiplexers 1250c outputs new path metrics on Lower Path Metric Bus 1295 and Upper Path Metric Bus 1296. The buses 1295 and 1296 are connected to LogLikelihood processors 1250a and 1250b, a bank of multiplexers 1278a and a Forward Address Processor 1290.

[0049] The Forward Address Processor 1290 receives a Forward Trellis Select 1232, a Forward Trellis Hold 1234, a Forward Trellis Transparent Bit 1236 and a Path Metric Input MUX Select 1238 from the control unit 1210, which are used to configure the Forward Address Processor 1290 in accordance with whether the unified decoder 1200 is being used to navigate a trellis in the forward or reverse direction.

[0050] The Forward Address Processor 1290 orders the new path metrics received on buses 1295 and 1296 such that an apparently sequential list of path metrics is presented to the butterfly processor 1260 for computation of the next column of the trellis, when the trellis is being navigated in the forward direction. When a trellis is being navigated in the reverse direction, the Forward Address Processor 1290 acts transparently.

[0051] The Path Metric Store 1280 receives addressing information ADDR0 1228a and ADDR1 1228b, Path Metric Reset 1230 and Path Metric Read/Write Clock 1231 from the control unit 1210, in addition to forward trellis path metrics 1285, which are output from the Forward Address Processor 1290. The Path Metric Store 1280 outputs stored path metrics 1276 to a bank of multiplexers 1278a and to LogLikelihood Processors 1250a and 1250b.

[0052] The bank of multiplexers 1278a is used as an interconnect point for multiple decoder row configurations, and receives stored path metrics 1276, a control signal 1278b from the control unit 1210, and new path metrics on buses 1295 and 1296. The bank of multiplexers 1278a allows the initialisation of the beta computation during LogMAP calculation and produces an output 1277 to Normalisation Subtractors 1278.

[0053] A comparator 1247 receives the butterfly path metrics output on buses 1266 and 1267 from the Butterfly Decoding Processors 1260 and determines a maximum new path metric. This maximum new path metric is then compared with a stored maximum path metric and the greater of the two values is presented as normalising output 1246, which is sent to the Normalisation Subtractors 1278 and the Intermediate Decoding Memory and Processor 1240.

[0054] The Normalisation Subtractors 1278 receive the output 1277 from the bank of multiplexers 1278a and subtract the Normalising Output 1246 to ensure that the path metrics are contained within the dynamic range of the architecture.

The normalised path metrics 1275 are output and presented to a Reverse Address Processor 1270 and LogLikelihood Processors 1250a and 1250b. The Reverse Address Processor 1270 also receives as inputs LogLikelihood Enable 1214, LogLikelihood 0 Enable 1203_0 and LogLikelihood 1 Enable 1203_1, Reverse Trellis Select 1222, a Reverse Trellis Hold 1224 and a Reverse Trellis Transparent Bit 1226 from the control unit 1210. The inputs from the control unit 1210 are used to configure the Reverse Address Processor 1270 appropriately, depending on whether the decoder 1200 is traversing a trellis in the forward or reverse direction. The output of the Reverse Address Processor 1270 is presented as reverse trellis path metrics 1265 to the Butterfly Decoding Processors 1260.

[0055] The Reverse Address Processor 1270 orders the normalised path metrics such that a desired sequence of path metrics is presented to the butterfly processor 1260 for computation of the next column of the trellis, when the trellis is being navigated in the reverse direction. When the trellis is being navigated in the forward direction, the Reverse Address Processor 1270 acts transparently.

[0056] The LogLikelihood Processor 1250a receives a LogLikelihood Mode 1214a, reverse trellis hold 1224a, reverse trellis transparent bit 1226a, a LogLikelihood rate 1248a, a LogLikelihood constraint 1249a, a LogLikelihood clock 1251a, a LogLikelihood reset 1252a, LogLikelihood polynomials 1253a, LogLikelihood 0 Enable $1203a_0$, LogLikelihood Enable $1203a_1$, reverse trellis select 1222a, and select signal 1258a from the control unit 1210. The LogLikelihood Processor 1250a also receives as inputs the normalised path metrics 1275, the output 1291b from the Input Symbol History 1298, stored path metrics 1276, new path metrics on buses 1296 and 1295 and interleaver extrinsic information 1256. The LogLikelihood processor 1250a produces a first output 1245a, which is presented to a LogLikelihood Ratio Processor 1297. The LogLikelihood processor 1250a also presents inputs 1266' and 1267' to the bank of multiplexers 1250c.

[0057] A second LogLikelihood processor 1250b receives corresponding inputs 1214b, 1224b, 1226b, 1248b, 1249b, 1251b, 1252b, 1253b, $1203b_0$, $1203b_1$, 1222b and 1258b from the control unit 1210. The LogLikelihood Processor 1250b also receives as inputs the normalised path metrics 1275, stored path metrics 1276, interleaver extrinsic information 1256 and the new path metrics on buses 1296 and 1295. The LogLikelihood processor 1250b produces an output 1245b, which is presented to the LogLikelihood Ratio Processor 1297.

[0058] The LogLikelihood Processor 1250a is used to compute dummy betas in the first phase of a LogMAP calculation. In the second phase of the LogMAP calculation, LogLikelihood Processors 1250a and 1250b are used in conjunction with the Butterfly Decoding Processors 1260 to create a LogLikelihood result for a "1" and a "0", respectively.

[0059] The Intermediate Decoding Memory and Processor 1240 acts as a buffer for producing output during a Viterbi computation. During a LogMAP computation, the Intermediate Decoding Memory and Processor 1240 acts as an extended store for the path metric store 1280. The Intermediate Decoding Memory and Processor 1240 receives an Intermediate Decoding Mode 1212, an Intermediate Decoding Direction 1237, a Spreading Input 1243, read/write clock 1257, a reset 1259 and a clocking signal 1254 from the control unit 1210. The Intermediate Decoding Memory and Processor 1240 also receives the Normalising Output 1246 and Decision Bits 1255. The Intermediate Decoding Memory and Processor 1240 produces extrinsic information 1242 and Traceback processor output 1567 to the LogLikelihood Ratio Processor 1297, and receives an input 1293 from the LogLikelihood Ratio Processor 1297. The Intermediate Decoding Memory and Processor 1240 also produces interleaver extrinsic information 1256 to LogLikelihood Processors 1250a and 1250b.

[0060] The LogLikelihood Ratio Processor 1297 receives a Hard or Soft Output Select 1213 and Spreading Input 1243 from the control unit 1210 in addition to the outputs 1245a and 1245b from the LogLikelihood Processors 1250a and 1250b. The LogLikelihood Ratio Processor 1297 also receives as inputs the extrinsic information 1242 of the Intermediate Decoding Memory and Processor 1240 and Scramble Address Data 1286. The LogLikelihood Ratio Processor 1297 then produces a Decoded Output 1294 and an output 1293 to the Intermediate Decoding Memory and Processor 1240.

[0061] The outputs 1245a and 1245b represent the probability of the decoded output being a "1" or a "0", respectively. The LogLikelihood Ratio Processor 1297 performs a subtraction of the outputs 1245a and 1245b in the log domain, which is equivalent to performing a division in the natural number domain. The result of the subtraction provides the Decoded Output 1294. The LogLikelihood Ratio Processor 1297 also subtracts the outputs 1245a and 1245b and the extrinsic information 1242 to produce the output 1293, which represents new extrinsic information.

[0062] A code of maximum constraint length k produces a trellis diagram with 2^{k-1} states. Fig. 6A shows a 32-state raw trellis diagram 1000, corresponding to a code having a maximum constraint length of 6. Each of the 32 states 1002 at time S_t has two possible branch metrics mapping to one of 32 states 1004 at time S_{t+1}. For example, state 0 1003 at time S_t has branch metrics 1006 and 1008 leading to state 0 1009 and state 16 1007 at time S_{t+1}.

[0063] The 32-state raw trellis diagram 1000 may be represented by 16 corresponding butterfly connections 1010 of the same trellis. It can be seen that pairs of states in one column 1012 of the trellis map to corresponding pairs of states in another column 1014 of the trellis. The trellis states 1014 at time S_{t+1} represent resultant path metrics. Each of the butterfly connections 1010 may be processed by a single butterfly processor 1260. In accordance with a preferred embodiment of the invention, as shown in Fig. 4, four butterfly processors 1260 are provided. This allows 8 resultant

path metric locations to be calculated in each clock cycle.

[0064] Fig. 6B shows the resultant path metric locations 1014 for a 32-state trellis diagram. The 32 resultant path metric locations have been ordered into four columns 1022, 1024, 1026 and 1028, each of which contains eight resultant path metric locations produced by four butterfly processors.

[0065] A trellis operation incorporates several sub-trellis operations, each of which corresponds to a single clock cycle. Figs. 7A, 7B, 7C, 7D and 7E show the process by which a preferred embodiment of the invention implements in-place path metric addressing. Fig. 7A shows time t=1, corresponding to the first sub-trellis operation, in which eight new path metrics 1112 are presented as inputs. New path metrics 1112 representing 0, 1, 2 and 3 are written into a first column of memory 1102, corresponding to upper memory blocks B0 of Path Metric Store 1280, whilst path metrics 16, 17, 18 and 19 are written into four holding registers 1114. Path metrics 16, 17, 18 and 19 are held for a clock cycle before being written to memory, as the memory locations to which they will be written will not become available until the next clock cycle when the new path metrics for trellis states 8 to 15 have been calculated.

[0066] In the next clock cycle t=2, shown in Fig. 7B, a further eight new path metrics 1122 are presented as inputs. The path metrics 1122 corresponding to new path metric locations 4, 5, 6 and 7 are written into a first column of memory 1104, corresponding to lower memory blocks B1 of Path Metric Store 1280. The contents of the holding registers 1114 are written into a second column of memory 1102, corresponding to Path Metric Store 1280 B0 and the new path metrics corresponding to path metric locations 20, 21, 22 and 23 are written in as the new contents of the holding registers 1114.

[0067] In the third clock cycle shown in Fig. 7C, a further group of new path metrics 1134 is presented. The new path metrics corresponding to states 8, 9, 10 and 11 are written into a third column of memory 1102, corresponding to Path Metric Store 1280 B0 and the contents of the holding registers 1114, being states 20, 21, 22 and 23, are written into a second column of a memory 1104, corresponding to Path Metric Store 1280 B1. The four new path metrics corresponding to states 24, 25, 26 and 27 are written into the holding registers 1114.

[0068] Fig. 7D shows the fourth clock cycle, during which the final eight new path metrics 1144 are presented. The new path metrics corresponding to states 12, 13, 14 and 15 are written to a third column of memory 1104, corresponding to Path Metric Store 1280 B1, the contents of the holding register corresponding to states 24, 25, 26 and 27 are written to a fourth column of memory 1102, corresponding to Path Metric Store 1280 B0. and the new path metrics corresponding to states 28, 29, 30 and 31 are written to holding registers 1114.

[0069] An additional clock cycle corresponding to t=5, as shown in Fig. 7E, is required to write the contents of the holding registers 1114 into a fourth column of memory 1104, corresponding to Path Metric Store 1280 B1.

[0070] Fig. 7F shows a representation of the addressing of the path metric columns for a 32-state trellis in accordance with a preferred embodiment of the present invention. The addressing sequence of the path metric columns 1150 corresponds to the read/write addresses of the path metric columns. Each row of the table 1160 corresponds to a different column of a trellis diagram, representing Symbol time n (S_n), Symbol time n+1 (S_{n+1}) and Symbol time n+2 (S_{n+2}). It is evident that the movement of the addresses of the path metric columns is periodic.

[0071] Figs. 7A-E have shown the progression from S_n to S_{n+1}. The next clock cycle, t=6, will begin the transition from S_{n+1} to S_{n+2} and columns 0, 2, 1, 3 will be executed in order to present a sequential list of states to the ACS units.

[0072] Figs. 8A, 8B, 8C, 8D, 8E and 8F show the process by which a preferred embodiment of the invention implements in-place path metric addressing during navigation of a reverse trellis. Fig. 8A depicts the notation that will be followed in Figs. 8B-F. Fig. 8B shows time t=1, corresponding to the first sub-trellis operation. The path metrics resident in the first column of memory A, $C_{0A,}$ have been shifted to a hold register 3010. In Fig. 8C, time t=2, the first column of memory B, $C_{0B,}$ is moved to the hold register 3010 and a function of C_{OA} and C_{2A} form resultant path metrics C_{OA}' and C_{0B}', which are written into the first column of memories A and B respectively. In Fig. 8D, the second column of memory A C_{1A} is deposited in the hold register 3010. A function of the previous contents of the hold register C_{0B} and C_{2B} form new path metrics C_{1A}' and C_{1B}' which are written back into the third column of A and B respectively.

[0073] Fig. 8E shows time t=4 in which C_{1B} is written to the hold register 3010. A function of C_{1A} and C_{3A} produces new path metrics C_{2A}' and C_{2B}', which are written into the second columns of memories A and B respectively. Fig. 8F shows the reverse sub-trellis operation corresponding to time t=5 in which a function of C_{1B} and C_{3B} forms resultant path metrics C_{3A}' and C_{3B}' which are written into fourth columns of memories A and B respectively. In the calculations of the reverse trellis, path metrics are presented to four butterfly processors in a scrambled manner and the in-place path metric addressing described in Figs 8B to 8F ensures that the resultant path metrics are presented in a sequential manner.

[0074] Fig. 9A shows a high level schematic block diagram representation of an embodiment of the Intermediate Decoding Memory and Processor 1240, which performs traceback and interleaver functions in respective decoding schemes. The Intermediate Decoding Memory and Processor 1240 receives as inputs decision bits 1255, normalising output 1246, spreading input 1243, intermediate decoding direction 1237, intermediate decoding mode 1212, a clocking signal 1254, a read/write clock 1257, a reset signal 1259 and the output 1293 from the LogLikelihood processor 1297. The Intermediate Decoding Memory and Processor 1240 produces outputs including extrinsic information 1242, inter-

leaver extrinsic information 1256 and Traceback processor output 1567.

[0075] Fig. 9B shows an exploded view of the Intermediate Decoding Memory and Processor 1240. A Traceback Address Controller 1510 receives as inputs Decision Bits 1255, the Intermediate Decoding Direction 1237, Normalising Output 1246, the clocking signal 1254, reset signal 1259, read/write clock 1257 and the Intermediate Decoding Mode 1212, which is inverted. The Traceback Address Controller 1510 produces an output 1567.

[0076] The Traceback Address Controller 1510 writes Decision Bits 1255 to a Window Memory Subsystem 1520 every clock cycle. During traceback, the Traceback Address Controller 1510 examines a trellis section to determine a biggest value to be used as a starting point. It is to be noted that it is not necessary to store the complete value for each state as a new traceback byte address can be generated using one of the Decision Bits 1255.

[0077] An Interleaver Controller 1520 also receives a clocking signal 1254, reset signal 1259, read/write clock 1257 and Intermediate Decoding Mode 1212. In addition, the Interleaver Controller 1520 receives the output 1293 from the LogLikelihood Ratio Processor 1297, the Spreading Input 1243 and the Intermediate Decoding Direction 1237. The Interleaver Controller 1520 produces extrinsic information 1242 and 1256. The extrinsic data 1242 is used as a recursive input to the Butterfly Processors 1260 when the decoder 1200 operates as a Turbo decoder.

[0078] The Interleaver Controller 1520 produces extrinsic information 1242 and 1256 at the beginning of every clock cycle. At the end of every clock cycle, the Interleaver Controller 1520 receives new extrinsic information in the form of the output 1293 from the LogLikelihood Ratio Processor 1297 and writes it into memory.

[0079] The Traceback Address Controller 1510 and Interleaver Controller 1520 are interconnected and supply a joint read/write signal 1515 to a Window Memory Subsystem 1530. The Traceback Address Controller 1510, Interleaver Controller 1520 and Window Memory Subsystem 1530 are further interconnected by a bi-directional data bus 1526 and an address bus 1525. The Interleaver Controller 1520 has a second address bus 1535 connected to the Window Memory Subsystem 1530 and the Window Memory Subsystem 1530 produces an output on a second data bus 1536 to the Interleaver Controller 1520.

[0080] Fig. 9C shows the Traceback Processor 1510. The decision bits 1255 are presented to a first multiplexer 1550. The output of the multiplexer 1550 is presented to a decisions register 1555. The output of the decisions register 1555 is data 1526, which is presented as an output of the Traceback Processor 1510 and is also fed back as a recursive input of the first multiplexer 1550 and as an input to a bit select 1558.

[0081] The Intermediate Decoding Direction 1237 is presented as an input to an address translation unit 1560. The address translation unit 1560 also receives a read/write clock 1257 and produces an output address 1525 and a read/write signal 1515. The read/write clock 1257 is also presented as the select of the first multiplexer 1550.

[0082] A normalising output 1246 is presented as an input to a state register 1562. The output of the state register 1562 is presented as an input to the address translation unit 1560, as well as being an input to a previous state unit 1564. The previous state unit 1564 presents two inputs to a second multiplexer 1566, whose output is the Traceback Processor Output 1567.

[0083] The output of the bit select 1558 is presented as an input to a first AND gate 1568. The output of the AND gate 1568 is presented as an input to the state register 1562. The output of the bit select 1558 is also presented to a second AND gate 1569, whose output is also presented to the state register 1562.

[0084] The Intermediate Decoding Direction 1237 is presented as the second input to the first AND gate 1568 and as the select input of the multiplexer 1566. The decode unit output 1502 is also presented, via a NOT gate 1570, to the second AND gate 1569.

[0085] Fig. 9D shows the interleaver controller of 1520 of Fig. 9B. The Intermediate decoding mode 1212 is presented to an AND gate 1580, whose output is presented to two tri-state buffers 1582 and 1583. The other input to the AND gate 1580 is the inverted form of the spreading input 1243. The tri-state buffer 1582 also receives as a input a read/write clock 1257. The second tri-state buffer 1583 receives the output 1293 from the LogLikelihood ratio processor 1297 as its second input. The output 1293 from the LogLikelihood Ratio Processor 1297 is also presented as inputs to two logic blocks 1584 and 1586. The spreading input 1243 is presented to each of the logic blocks 1584 and 1586, as is the reset signal 1259, and the clock signal 1254. The Interleaver 1520 receives data bus 1526 as an input and presents a corresponding output being extrinsic information 1242. A second data bus 1536 is output as interleaver extrinsic information 1256. The data bus 1526 is bi-directional, and the output of the Interleaver 1520 to the data bus 1526 is the output of the tri-state buffer 1583.

[0086] The first logic block 1584 receives the intermediate decoding mode 1212 and the intermediate decoding direction 1237 and produces an address 1525. The second logic block 1586 also receives the intermediate decoding mode 1212 and intermediate decoding direction 1237 and produces address 1535. Each of the logic blocks 1584 and 1586 also receive an input beta_d, which is a low or high power signal.

[0087] Fig. 9E shows an exploded view of the logic block 1584 of Figure 9D. The logic block 1586 of Figure 9D has the same configuration. A window count 1590 receives as inputs a reset 1259, a clock 1254 and an enable 1212. It also receives as an input the output of a first adder 1592. The window count 1590 produces an output which is presented to adders 1592 and 1593. The first adder 1592 receives as a second input the constant 1599 and presents its output

to the window count 1590. The bit count 1591 receives as inputs the reset 1259, the clock 1254, the enable 1212 and the output of a third adder 1594. The bit count 1591 produces an output which is presented to two adders 1593 and 1594. Beta_d is presented to an element 1595, which adds 1 and if beta_d is active, it negates the value and presents a result as a second input to the third adder 1594. The output of the adder 1594 is presented as a recursive input to the bit count 1591.

[0088] The output of the second adder 1593 is presented as an input to a multiplexer 1596 and to a scramble 1597. The multiplexer 1596 receives a select signal indicating if the architecture is operating as a first or second decoder, and a second input being the output of the scramble 1597. The output of the multiplexer 1596 is the address 1525. The scramble 1597 receives the Spreading Input 1243 as an enabling signal and the output 1293 from the LogLikelihood Ratio Processor 1297 as data. The scramble 1597 could be memory or logic function as is well known in the art and is used to implement scrambling of addresses between a first and second decoder when undertaking Turbo decoder calculations.

[0089] Fig. 9F shows a schematic block diagram representation of the window memory sub system 1530 of Fig. 9A. A read/write clock 1515, address buses 1525 and 1535, and data buses 1526 and 1536 are presented to a window address decoder 1530a and window memories 1530b...1530d.

[0090] Fig. 10A shows the LogLikelihood Processor 1250a of Fig. 4. A bank of four butterfly units 1410 is provided and its constituent ACS units 1412a...1412h are presented with pairs of reverse trellis path metrics 1415a...h from a Reverse Address Processor 1270b and stored path metrics 1276 from the Path Metric Store 1280. The stored path metrics 1276 represent alphas in the LogMAP calculation. Each of the ACS units 1412a...h is also presented with a LogLikelihood Mode 1214a, a LogLikelihood Clock 1251a and a LogLikelihood Reset 1252a. BMC units 1414a...1414d are each provided with a number of inputs, including the LogLikelihood Rate 1248a, the LogLikelihood Constraint 1249a, the LogLikelihood Polynomials 1253a, interleaver extrinsic information 1256 and the Input Symbol History 1291b. The ACS units 1412a...h produce first outputs 1413a...1413h, which are presented in sequential pairs to the ACS node units 1420a...1420d. The ACS units 1412a...h produce second outputs 480a...h, each of which is presented to a corresponding normalising subtractor 1470a...1470h. The normalising subtractors 1470a...1470h produce outputs 1266' and 1267', which are fed recursively via multiplexers, as explained below, to the Reverse Address Processor 1270b and used to ensure that the path metrics remain within the dynamic range of the architecture.

[0091] Each of a first bank of multiplexers 1417a...h receives a corresponding normalised path metric 1275a...h from the Normalising Processor 1278 and a select signal 1258 from the control unit 1210. Multiplexers 1417a...d also receive corresponding path metrics 1296a...d, and multiplexers 1417e...h receive corresponding path metrics 1295a...d. The path metrics 1295a...d and 1296a...d represent betas in the LogMAP calculation. The select signal 1258 is used to determine whether the normalised path metrics 1275a...h or the path metrics 1295a...d and 1296a...d will be output.

[0092] Each of a second bank of multiplexers 1416a...h receives LogLikelihood Mode 1214a as a select signal and a corresponding output from the first bank of multiplexers 1417a...h. Multiplexers 1416a...d receive a third input, being the output 1266' of the normalising subtractors 1470a..d and multiplexers 1416e..h receive the output 1267' from the normalising subtractors 1470e...h. The outputs from the multiplexers 1416a..h are presented as inputs to the Reverse Address Processor 1270b.

[0093] The Reverse Address Processor 1270b also receives a LogLikelihood Mode 1214a, Turbo enable for LogLikelihood 0 Enable $1203a_0$, Turbo enable for LogLikelihood 1 $1203a_1$, reverse trellis selector 1222a, reverse trellis transparent bit 1226a and the reverse trellis hold 1224a. The beta outputs 1266' and 1267' of the LogLikelihood Processor 1250a represent the final dummy beta values used for the start of the beta processing phase, when the decoder 1200 is operating in LogMAP/turbo mode.

[0094] The outputs of the ACS node units 1420a and 1420b are presented to an ACS node unit 1430a and the outputs of the ACS node units 1420c and 1420d are presented to an ACS node unit 1430b. The outputs of the ACS node units 1430a, 1430b are presented as inputs to a further ACS node unit 1440a, whose output is presented to a multi-row comparator tree, which spans the decoder when operated in a multi-row configuration so as to capture the maximum path metric being calculated for the state of the trellis being investigated. An output from the multi-row comparator tree is presented to a subtractor 1450 and a register 1460. The subtractor 1450 also presents a recursive input to the register 1460. The register output 1245a is fed to the subtractor 1450 and to each one of the normalising subtractors 1470a...1470h, in addition to being an output of the LogLikelihood Processor 1250a.

[0095] Fig. 10B shows one arrangement of the ACS node unit 1420a of Fig. 10A. The outputs 1413a and 1413b from the ACS leaf units are presented as inputs to a comparator 1474 and a multiplexer 1476. A borrow output of the comparator 1474 is fed as a select signal of the multiplexer 1476. A difference output of the comparator 1474 is presented as an input to a log sum correction table 1478. The output of the log sum correction table 1478 is presented to an adder 1480, whose second input is the output of the multiplexer 1476. The adder 1480 computes and outputs the sum 1425a of its two inputs, the sum 1425a representing the maximum of the two inputs 1413a and 1413b, with a log-sum correction.

[0096] Fig. 10C shows a configuration of a LogLikelihood processor 1250a of Fig. 4 for an eight row decoder em-

bodiment. The LogLikelihood processors 1250a' in each of the rows are interconnected via a bank of multiplexers 1490. Each multiplexer 1490 presents a single input to a LogLikelihood processor 1250a' in its corresponding decoder row. Pairs of LogLikelihood processors 1250a' present their outputs as inputs to ACS node units 1420a', 1420b', 1420c' and 1420d'. The outputs of the LogLikelihood processors 1250a' are also presented as recursive inputs to the bank of multiplexers 1490. The ACS nodes units 1420a', 1420b', 1420c' and 1420d' are paired and present their outputs as inputs to further ACS nodes units 1430a' and 1430b'. The outputs of the ACS node units 1420a', 1420b', 1420c' and 1420d' are also presented as recursive inputs to the bank of multiplexers 1490. The ACS node units 1430a' present their outputs to a final ACS node unit 1440' and as recursive inputs to the bank of multiplexers 1490. The output of the final ACS node unit 1440' is presented as a final recursive input to the bank of multiplexers 1490. Each multiplexer 1490 is presented with a select signal.

[0097] Fig. 10D shows one useful architecture of an ACS unit 1412a of Fig. 10A. A first pair of inputs, branch metric 1 402' and branch metric 0 406', are presented to a multiplexer 408, which produces an output 402". A second pair of inputs, path metric 1276a and path metric 1 1415b are presented to a multiplexer 409, which produces an output 403'. Each of the multiplexers 408 and 409 receives LogLikelihood Mode 1214a as a select signal. When the LogLikelihood Mode 1214a is inactive, branch metric 1 402' is selected by multiplexer 408 and path metric 1 1415b is selected by multiplexer 409. Conversely, when LogLikelihood Mode 1214a is active, branch metric 0 406' is selected by multiplexer 408 and path metric 1276a, representing an alpha value, is selected by 409.

[0098] The outputs 402" and 403' of the multiplexers 408 and 409 are presented to an adder 410'. The sum 411' is output from the adder 410' and presented to a multiplexer 416' and a multiplexer 417'. The multiplexer 417' receives branch metrio 0 406' as a second input and LogLikelihood Mode 1214a as a select signal. The output 418' of the multiplexer 417' is presented to an adder 412'. The adder 412' receives path metric 0 1415a as a second input. The adder 412' produces a sum 413', which represents the sum of alphas, betas and gammas. The sum 413' is presented to a Full Subtractor 414' and a multiplexer 420'. The multiplexer 416'receives a hardwired input 407' corresponding to the minimum 2s complement number able to be represented and a LogLikelihood Mode 1214a as a select signal. The Full Subtractor 414' also receives the output 408' of the multiplexer 416' as a second input and produces a borrow 361' and a difference 415'.

[0099] The output 408' of the multiplexer 416' is presented as a first input to a multiplexer 420'. The multiplexer 420' receives the sum 413' of the adder 412' as a second input. The borrow output 361' of the Full Subtractor 414' is fed to the multiplexer 420' to compute a maximum MAX of the input values. A second output 415' of the Full Subtractor 414', representing the difference of the multiplexer output 408' and the sum 413', is presented to a Log-sum correction table 440', which tweaks the result of the new path metric, when the output of the Full Subtractor 414' is small, to produce a more accurate result in the Log domain for LogMAP decoding. An output 441' of the Log-sum correction table 440' is presented to an Adder 460'. An output 421' of the multiplexer 420' is also presented to the Adder 460'. A result 490' from the Adder 460' is then presented as an input to an accumulate register 470'. The accumulate register 470' accumulates values for dummy beta LogMAP calculations. The output 480a of the accumulate register is presented as an input to a further multiplexer 475' and as an output of the ACS unit 1412a to be used in dummy beta computation. The multiplexer 475' receives the sum 490' as a second input and LogLikelihood Mode 1214a as a select signal. The output 1413a of the multiplexer 475' is the second output of the ACS unit 1412a.

[0100] Fig. 11 shows Butterfly Decoding Processors 1260 of Fig. 4 in accordance with a preferred embodiment. Each of the component ACS units ACS0...ACS7 is presented with a number of inputs, including a butterfly mode 1221, a butterfly reset 1215, a butterfly clock 1217 and a beta-phase enable 1235. Each of the component BMC units BMC0... BMC3 is presented with a butterfly rate 1216, a butterfly constraint 1220 and butterfly polynomials 1218. The reverse trellis path metrics 1265 fan out to present inputs 1265a...1265h to the ACS units ACS0...ACS7, such that each ACS unit receives two reverse trellis path metrics. Reverse trellis path metrics 1265a and 1265b are presented to each of the ACS units ACS0 and ACS1, reverse trellis path metrics 1265c and 1265d are presented to each of the ACS units ACS2 and ACS3, reverse trellis path metrics 1265e and 1265f are presented to each of the ACS units ACS4 and ACS5, and reverse trellis path metrics 1265g and 1265h are presented to each of the ACS units ACS6 and ACS7. The BMC units BMC0...BMC3 also receive as inputs extrinsic information 1242 and input symbol history input symbol 1291a.

[0101] The Butterfly Decoding Processors 1260 are preferably formed by eight ACS units and four BMCs, configured as four butterfly processors:

(i)ACS0, BMC0, ACS1;
(ii)ACS2, BMC1, ACS3;
(iii)ACS4, BMC2, ACS5; and
(iv) ACS6, BMC3, ACS7.

[0102] The unified decoder architecture takes advantage of the fact that each state in a trellis diagram may only be impacted upon by two other states. A code with a minimum constraint length of k gives rise to a trellis diagram having

2^{k-1} states. A butterfly processor having two ACS units and an intermediary BMC unit is capable of processing two states in a trellis state diagram. Therefore, in order to process a code with constraint length 4 in one clock cycle, a total of eight ACS units are required. More states may be handled by processing over a greater number of clock cycles, or by having more butterfly processors.

[0103] The ACS units ACS0...ACS7 produce corresponding outputs 1255a...h, which are aggregated to form decision bits 1255. New path metrics computed by ACS units ACS0...ACS3 are presented as outputs 1267a...d and sent on upper new path metric bus 1267. The new path metrics 1266a...d calculated by ACS units ACS4...7 are presented to the lower new path metric bus 1266.

[0104] Fig. 12 shows the Reverse Address Processor 1270 of Fig. 4. The Reverse Address Processor 1270 provides facilities for delaying and ordering path metrics to produce a desired pattern of path metrics. The Reverse Address Processor 1270 is also capable of acting transparently when the decoder 1200 is operating in forward trellis mode such that input path metrics are presented as outputs without alteration. The Reverse Address Processor 1270 receives as inputs: a reverse trellis selector 1222, a reverse trellis hold 1224, a reverse trellis transparent bit 1226, a LogLikelihood Mode 1214, a LogLikelihood 0 Enable 1203_0, a LogLikelihood 1 Enable 1203_1, and the normalised path metrics 1275. The normalised path metrics 1275 fan out to present pairs of inputs 1275a and 1275e. 1275b and 1275f, 1275c and 1275g, and 1275d and 1275h to a first bank of corresponding multiplexers 1910a3, 1910b3, 1910c3 and 1910d3, and a second bank of corresponding multiplexers 1915a...1915d.

[0105] The reverse trellis selector 1222 is presented to each of the first bank of XOR gates 1920a...d. The XOR gates 1920a and 1920c receive LogLikelihood 0 Enable 1203_0 and XOR gates 1920b and 1920d receive LogLikelihood 1 Enable 1203_1. Each XOR gate 1920a...d produces an output which is presented to a corresponding XOR gate in a second bank of XOR gates 1925a...d and to a corresponding one of the multiplexers 1910a3...1910d3. Each of the second bank of XOR gates 1925a...d receives LogLikelihood Enable 1214 as a second input and produces an output to a corresponding multiplexer in the second bank of multiplexers 1915a...d. As mentioned above, each multiplexer 1915a...d receives a pair of normalised path metrics The outputs from the XOR gates 1925a...d act as select signals for the respective multiplexers 1915a...d to choose one of the presented normalised path metrics. Each of the multiplexers 1915a...d presents an output to a corresponding one of multiplexers 1910b1, 1910d1, 1910f1 and 1910h1.

[0106] Multiplexers 1910a3...d3 an 1915a...d are presented with different pairs of inputs depending on the values of the LogLikelihood Enable 1214 and the LogLikelihood Enable 0 1203_0 and LogLikelihood 1 Enable 1203_1. LogLikelihood 0 Enable 1203_0 is enabled for LogLikelihood Processor 0 and disabled for LogLikelihood Processor 1. Conversely, LogLikelihood 1 Enable 1203_1 is enabled for LogLikelihood Processor 1 and disabled for LogLikelihood Processor 0. As the Reverse Address Processor 1270 is used in several locations within the unified decoder 1200, the Reverse Address Processor 1270 must be capable of handling different modes of operation. When the LogLikelihood Enable 1214 and LogLikelihood Enables 1203_0 and 1203_1 are inactive, the Reverse Address Processor 1270 is in Viterbi mode operating on a trellis generated by a non-systematic convolutional code. When LogLikelihood Enable 1214 is active, the Reverse Address Processor 1270 is performing reverse trellis switching for the LogLikelihood operation for LogMAP decoding. When either of the LogLikelihood Enables 1203_0 and 1203_1 is active with the LogLikelihood Enable 1214 active, the Reverse Address Processor 1270 is performing switching appropriate for a LogLikelihood operation using a recursive systematic code, as in Turbo decoding. The XOR gates implement the appropriate switching for the different operating modes of the Reverse Address Processor 1270.

[0107] Each of the first bank of multiplexers 1910a3...1910d3 produces an output which is presented to a corresponding latch 1910a2...1910d2. Each of the latches 1910a2...1910d2 receives the reverse trellis hold 1224 as an input and presents a delayed output as the second input to a corresponding one of the multiplexers 1910a1, 1910c1, 1910e1 and 1910g1.

[0108] The reverse trellis transparent bit 1226 is broadcast to each of a third bank of multiplexers 1910a1... 1910h1, which produce corresponding path metrics 1265a...h. The path metrics 1265a...1265h are collated and presented as reverse trellis path metrics 1265, the output of the Reverse Address Processor 1270. When the decoder 1200 is operating in the forward trellis direction, the reverse trellis transparent bit 1226 is set such that the Reverse Address Processor 1270 allows the normalised path metrics 1275 to pass through to become the reverse trellis path metrics 1265, without alteration.

[0109] Fig. 13 shows Normalisation Subtractors 1278 of Fig. 4. The normalising output 1246 is presented as an input to each of the subtractors 1610a...1610h. The output 1277 of the bank of multiplexers 1278a is presented as individual path metrics 1277a...1277h, each of which is presented to corresponding subtractors 1610a...1610h. The outputs 1275a...h of the subtractors 1610a...1610h form the normalised path metrics 1275. The normalising subtractors 1278 are used to subtract the maximum path metric, calculated during the traversal of the trellis and presented as the normalising output 1246, from the new path metrics to ensure that the path metric values are retained within the dynamic range of the architecture.

[0110] Fig. 14 shows a comparator 1247 of Fig. 4, in accordance with a preferred embodiment of the invention. The butterfly path metrics presented on bus 1267 are fanned out to produce inputs 1267a...1267d to corresponding max-

imum comparators 1710a...1710d. Similarly, the butterfly path metrics presented on bus 1266 are fanned out to produce inputs 1266a...1266d to corresponding maximum comparators 1710e...1710h. The path metrics 1266a...1266d and 1267a...1267d are compared against one another and a maximum path metric 1715 is output to a multi-row comparator tree, shown in Fig. 17, which spans the decoder when operated in a multi-row configuration so as to capture the maximum path metric being calculated for the state of the trellis being investigated. An output 1716 from the multi-row comparator tree is presented to a register 1720, which stores the greatest path metric calculated during the traversal of the trellis. The output 1716 is also presented as an input to a subtractor 1730. The register 1720 provides a second input to the subtractor 1730, the input being the greatest path metric calculated thus far during the traversal of the trellis. The subtractor compares the greatest path metric calculated during the traversal of the trellis with the maximum path metric 1715 and if the maximum path metric 1715, which has just been calculated, is greater than the greatest path metric calculated during the traversal of the trellis, a load signal 1735 is enabled to the register 1720 so that the maximum path metric 1715 is loaded into the register 1720 to become the greatest path metric calculated during the traversal of the trellis. The register provides a further output, being a normalising output 1246, which is fed to the normalising subtractors 1278 and to the Intermediate Decoding Memory and Processor 1240. The normalising output 1246 is used to ensure that calculated path metric values remain within the dynamic range of the architecture.

[0111] Fig. 15 shows a path metric memory 1280 of Fig. 4, in accordance with a preferred embodiment. A path metric reset 1230 and path metric read/write clock 1231 are presented to each of the memory units 1810a...1810h. The upper memory blocks 1810a...1810d are clustered as B0, and receive an input ADDR0 1228a. Conversely, the lower memory blocks 1810e...1810h are clustered to form B1, and receive a corresponding input ADDR1 1228b. The path metric store 1280 receives the forward trellis path metrics 1285, which fan out and provide a path metric 1285a...1285h to each of corresponding memory blocks 1810a...1810h, as shown in the diagram. The path metric store 1280 buffers the forward trellis path metrics 1285 for one trellis processing cycle and then produces outputs 1276a...1276h, which are aggregated and form the stored path metrics 1276.

[0112] Fig. 16 shows a Forward Address Processor 1290 of Fig. 4, in accordance with a preferred embodiment of the present invention. The Forward Address Processor 1290 provides facilities for delaying and ordering path metrics to produce a desired pattern of path metrics. The Forward Address Processor 1290 is also capable of acting transparently when the decoder 1200 is operating in reverse trellis mode such that input path metrics are presented as outputs without alteration. The upper path metric bus 1296 is broken into its component path metrics 1296a...1296d, which are presented, as indicated, to two multiplexers 2010a and 2010b, each of the multiplexers receiving two input path metrics. The lower path metric bus 1295 is broken into its constituent path metrics 1295a...1295d and presented, as indicated, to two multiplexers 2010c and 2010d, each of the multiplexers receiving two input path metrics. The multiplexers 2010a...2010d each receive a forward trellis select 1232, which indicates which of the presented path metrics 1296a... 1296d and 1295a...1295d is to be selected.

[0113] Each of the multiplexers 2010a...2010d feeds into a corresponding hold register 2015a...2015d. The hold registers 2015a...2015d each receive an input, being forward trellis hold 1234. The purpose of the multiplexers 2010a... 2010d and the hold registers 2015a...2015d is to delay certain of the path metrics 1296a...1296d and 1295a...1295d by a clock cycle as part of the in-place path metric addressing.

[0114] Each of the hold registers 2015a...2015d produces an output which is presented to a bank of multiplexers 2020 as indicated. The other inputs to the bank of multiplexers 2020 are the constituent path metrics of the upper path metric bus 1296 and the lower path metric 1295, also as shown. A path metric input multiplexer select 1238 is broadcast to the bank of multiplexers 2020. The bank of multiplexers 2020 produces outputs to a second bank of multiplexers 2030, whose other inputs are the constituent path metrics of upper path metric bus 1296 and lower path metric bus 1295. A forward trellis transparent bit 1236 is provided to the second bank of multiplexers 2030 and is used to effect a transparent path when the decoder 1200 is operating in the reverse trellis mode. The bank of multiplexers 2030 produces path metrics 1285a...1285h, which are collated to form the forward trellis path metrics 1285, being the output of the Forward Address Processor 1290.

[0115] Fig. 17 shows the Comparator 1247 of Fig. 4, when used in an eight row decoder configuration. The comparators 1247' in each of the rows are interconnected via a bank of multiplexers 2110. Each multiplexer 2110 presents a single input 1716 to a corresponding comparator 1247' in its corresponding decoder row. Pairs of comparators 1247' present their outputs 1715 as inputs to ACS node units 1420a", 1420b", 1420c" and 1420d", each of which spans two rows of the decoder. The outputs 1715 of the comparators 1247' are also presented as recursive inputs to the bank of multiplexers 2110. The ACS nodes units 1420a", 1420b", 1420c" and 1420d" are paired and present their outputs as inputs to further ACS nodes units 1430a" and 1430b". The outputs of the ACS node units 1420a", 1420b", 1420c" and 1420d" are also presented as recursive inputs to the bank of multiplexers 2110. The ACS node units 1430a" present their outputs to a final ACS node unit 1440" and as recursive inputs to the bank of multiplexers 2110. The output of the final ACS node unit 1440" is presented as a final recursive input to the bank of multiplexers 2110. Each multiplexer 2110 is presented with a select signal.

[0116] Fig. 18 shows the configuration of the Input Symbol History 1298, including an address controller, of Fig. 4.

An Input Symbol History Address 1219 is presented as an input to a Window Decoder 2210, which decodes the address to enable access to a first double buffered memory bank-0 2216 and a second double buffered memory bank-1 2218. The Input Symbol History 1298 double buffers received input to ensure that a continuous data flow is maintained. Input Symbol History clock 1223 and Input Symbol History reset 1225 are presented to a counter 2212, whose output 1297a is also presented to the double buffered memory bank-0 2216 and the double buffered memory bank-1 2218. Input symbols 1299 are presented from a host processor to a demultiplexer 2214. The demultiplexer 2214 produces an output 2224 to double buffered memory bank-0 2216 and the second output 2226 to a double buffered memory bank-1 2218. The demultiplexer 2214 also receives as an input a read/write signal 1297b. The read/write signal 1297b also feeds a first multiplexer 2220 and a second multiplexer 2222. Each of the double buffered memory banks 2216 and 2218 is presented with a bank select signal 1211, with the bank select signal 1211 being inverted at the interface to double buffered memory bank-1 2218.

[0117] Double buffered memory bank-0 2216 produces a first output 2228 to the multiplexer 2220 and a second output 2230 to a second multiplexer 2222. Double buffered memory bank-1 2218 produces a corresponding first output 2232 which feeds multiplexer 2220 and a second output 2234 which is presented to the second multiplexer 2222. The first multiplexer 2220 produces an output 1291b which is presented as an input to LogLikelihood processor-0 1250b. The second multiplexer 2222 produces an output 1291a which is presented as an input to the butterfly processors 1260.

[0118] Fig. 18 also shows an exploded view of the double buffered memory bank-1 2218. Incoming data 2226 is presented to a 1-to-n demultiplexer 2240, which also receives a window select, being the output of the window decode 2210. N outputs from the demultiplexer 2240 are presented to n corresponding windows W0...Wn, each of which produces an output which is presented to a first m-to-1 multiplexer 2242 and a second m-to-1 multiplexer 2244. Each of the m-to-1 multiplexers 2242, 2244 also receives a window select input signal. The first m-to-1 multiplexer 2242 produces the output 2232 which is used for the dummy beta calculations and is destined for the LogLikelihood ratio processor-0 1250a. The second m-to-1 multiplexer 2244 produces the output 2234, which is used for calculating alphas and betas in the branch metric units of the butterfly processors 1260.

[0119] Fig. 19 shows the LogLikelihood ratio processor 1297 of Fig. 4. The LogLikelihood ratio processor 1297 receives inputs 1245a and 1245b, which are output from LogLikelihood processor-0 1250a and LogLikelihood processor-1 1250b, respectively. The LogLikelihood ratio processor 1297 also receives as inputs the extrinsic information 1242, the hard or soft output select 1213, Spreading Input 1243, the Traceback Process Output 1567 and Scramble Address Data 1286.

[0120] A subtractor 2310 receives the inputs 1245a and 1245b, representing the likelihood of a "1" and a "0", respectively, and produces an output 2315 which feeds a second subtractor 2320. The output 2315 of the subtractor 2310 also feeds a first multiplexer 2330 and forms part of an output 1294. The second input to the subtractor 2320 is the extrinsic information 1242. The output 2325 of the subtractor 2320 is presented to a second multiplexer 2340.

[0121] The Traceback Process Output 1567 is presented as a second input to the first multiplexer 2330. The hard or soft output select 1213 is presented as the select input of the multiplexer 2330 and the output of the multiplexer 2330 forms the zero bit of the decoded output 1294. The output 2315 of the subtractor 2310 is combined with the least significant bit of the output of the multiplexer 2330 to form a multi-bit decoded output 1294.

[0122] The second multiplexer 2340 receives Scramble Address Data 1286 as its second input and Spreading Input 1243 as its select signal. The second multiplexer 2340 produces an output 1293, which is fed from the LogLikelihood ratio processor 1297 to the Intermediate Decoding Result and Memory 1240.

[0123] The embodiment shown in Fig. 3 operates in a five-phase mode. As no loglikelihood processors are present, more path metric memory is required to store more alphas and betas in the computations performed by LogLikelihood Processors 1250a and 1250b in the embodiment of Fig. 4, which operates in a two-phase mode.

Operation

[0124] The first step in the operation of the decoder 1200 is to initialise the decoder such that the architecture embodies the required configuration of either convolutional decoding or turbo decoding. The variables available for manipulation include the number of columns needed for the trellis size in question, the number of states in the trellis, the mask for the appropriate number of bits to be used in the addressing of the columns in the path metric memory and the decision depth of the traceback process. The register which holds the winning path metric for the symbol being processed is initialised and sequential numbers are assigned to a register bank whose values are permuted between every symbol time to reflect the column address sequence required for each trellis operation.

[0125] It is to be noted that the decoder 1200 can operate in either the forward or reverse trellis direction.

[0126] In the case in which the trellis is being navigated in the forward direction, the Reverse Address Processor 1270 is configured to operate in transparent mode by setting the Reverse Trellis Transparent Bit 1226. When navigating the trellis in the forward direction, the sequential numbers are rotated to the left after their first use.

[0127] An iterative process begins by reading the path metrics from the column of the path metric store 1280 B0 and

B1 corresponding to the number of the iteration. The sequential list of path metrics held in the first column of 1280 B0 and 1280 B1 are presented to the butterfly processors 1260. The butterfly processors 1260 produce, via the bank of multiplexers 1250c, new path metrics, which are no longer in sequential destination state order and are fed into the Forward Address Processor 1290. The Forward Address Processor 1290 essentially performs a sort operation on each column of new path metrics with the resultant effect being that the columns in the path metrics memory 1280 B0 and B1 represent a set of sequential states when reading down the column. During each column operation, as shown in Figs 7A-E, half of the eight new path metrics are written directly into the path metric store 1280, whilst the remaining new path metrics are written into the hold registers 2015a...2015d within the Forward Address Processor 1290. This alternates between each group of path metrics.

[0128] The navigation through the forward trellis requires a number of column iterations, being one more than the number of columns needed for the particular trellis in question. If the number of iteration is even, path metrics from buses 1296A, C, E, G are written into the column of path metric store 1280 B0 corresponding to the number of the iteration. Path metrics from the buses 1296B, D, F, H are contemporaneously written into the hold registers 2015a... 2015d of the Forward Address Processor 1290.

[0129] If, however, it is an odd iteration, the path metrics from buses 1296A, C, E, G are written into the hold registers 2015a...2015d of the Forward Address Processor 1290 and path metrics from buses 1296B, D, F. H are written into the column of the path metric store 1280 corresponding to the number of the iteration.

[0130] During the column operations, the decision bits 1255 generated by the ACS units of the butterfly processor 1260 are grouped into a byte and written into the Intermediate Decoding Memory and Processor 1240. The next iteration in the process begins by reading the column address from the path metric store 1280 B0 and B1 corresponding to the number of the next iteration. The iterative process continues until the number of column iterations corresponds to one more than the number of columns required for the trellis being calculated.

[0131] A further write operation is required at the end of the iterative process to transfer the four new path metrics in the hold register of the Forward Address Processor 1290. The four new path metrics are written into the final column of path metric store memory 1280 B1. The final result is that the new path metrics have been written into path metric store 1280 B0 and B1, albeit in a different column order. However, it is to be noted that the order within each column has not changed.

[0132] When the trellis is being navigated in the reverse direction, the sequential numbers are rotated to the right and then used for the first time. A group of four path metrics are fetched from the first column of path metrics 1280 B1 and are placed in the holding registers within the Reverse Address Processor 1270. The Forward Address Processor 1290 is configured to operate in a transparent mode by setting the forward trellis transparent bit 1236. The corresponding reverse trellis transparent bit 1226 is set such that Reverse Address Processor 1270 is enabled. The navigation through the reverse trellis is described in Figs 8A-8F.

[0133] Navigating the trellis in the reverse direction requires a number of iterations corresponding to one more than the number of columns required for the particular trellis. When navigating the trellis in the reverse direction, the in-place path metric system always presents a scrambled list of path metrics through the Reverse Address Processor 1270 to produce a non-sequential list of path metrics to the butterfly processors 1260. The resultant trellis state ordering produced by the butterfly processors 1260 is trellis state sequential.

[0134] In the event that an even iteration is being undertaken, the column in the path metrics store 1280 B0 corresponding to the number of iterations plus one is read and passed through the multiplexers 1278a, normalising processors 1278 and the Reverse Address Processor 1270 to the butterfly processors 1260. The path metrics currently held in the Reverse Address Processor 1270 are also read into the butterfly processor 1260. The column in path metric store 1280 equivalent to the number of the iteration is read and written into the hold register of the Reverse Address Processor 1270.

[0135] In the case that the number of the iteration is odd, the column of path metric store 1280 B1 corresponding to the number of the iteration plus one is read and passed through the multiplexers 1278a and normalising processors 1278 to the Reverse Address Processor 1270 and then to the butterfly processor 1260. The path metrics held in the Reverse Address Processor 1270 are also presented as inputs to the butterfly processor 1260. The column of path metrics store 1280 B0 corresponding to the number of the iteration is read and written into the hold register of the Reverse Address Processor 1270.

[0136] At this point of the navigation of the reverse trellis, the sequential list of path metrics held in the first column of path metric stores 1280 B0 and B1 is presented to the Reverse Address Processor 1270. The Reverse Address Processor 1270 performs a sort operation on each column of new path metrics to the effect that the resultant columns presented to the butterfly processor 1260 are no longer in sequential destination state order. The butterfly processor 1260 produces eight new path metrics, which are presented, via a bank of multiplexers 1250c, to the Forward Address Processor 1290. The Forward Address Processor 1290 is in transparent mode , so the trellis-state sequential list of path metrics produced by the butterfly processors 1260, via the bank of multiplexers 1250c, is written back into the path metric stores 1280 B0 and B1. The path metrics stores 1280 B0 and B1 represent a set of sequential states when

reading down the column.

[0137] During the column operations, the decision bits 1255 generated by the ACS units of the butterfly processors 1260 are grouped into a byte and presented to the Intermediate Decoding Memory and Processor 1240. The next iteration commences by reading the appropriate column of path metrics from path metric stores 1280 B0 and B1.

[0138] At the conclusion of the iterative process, the new path metrics are back in path metrics store 1280 B0 and B1, albeit in a different column order. It is to be noted that the ordering within each column has not changed.

[0139] The traceback processor 1510 within the Intermediate Decoding Memory and Processor 1240 knows the trellis processing direction and the bit location of the decision bit as it performs the well known pointer based traceback operation. The decision bit is extracted from one byte and is used to generate the next pointer into the traceback memory 1530. Traceback terminates when a predefined traceback depth has been achieved. The traceback depth is typically between five and nine times the constraint length of the code.

[0140] When the decoder 1200 is being used for turbo decoding, the processing is broken into two distinct phases: dummy-beta/alpha processing and beta/LLR processing. When either the forward trellis or the reverse trellis operation is mentioned the above processing occurs, but only for the degenerate case of when the number of trellis states matches the number of ACS units ACS0...ACS7 in a multiple (power of 2) of the ACS unit size of the butterfly processors 1260. The LogLikelihood processor-0 1250a and the ACS units within the butterfly processor 1260 are each equipped with registers to allow the respective ACS units to accumulate results needed for alpha and beta calculations.

[0141] The calculation of dummy-betas and alphas occur in parallel. The LogLikelihood processor-0 1250a performs a dummy beta calculation using the leaf ACS units at its disposal. This calculation requires access to the input symbol history buffer and the Intermediate Decoding Memory and Processor interleaver memory, each of which is a windowed memory system. The input symbol history buffer is organised into banks 2216 and 2218 of the size of a processing window. The LogLikelihood processor-0 1250a accumulates dummy betas by processing at time t the window to be processed at time t+1. The LogLikelihood processor-0 1250a does not need to access the path metric stores 1280, which is why the LogLikelihood ratio processor-0 1250a can operate in parallel to the ACS units contained within the Butterfly processors 1260.

[0142] The LogLikelihood ratio processor-0 1250a performs normalisation on the dummy beta values by using the adders in the ACS tree to determine the maximum beta calculated. This value is then subtracted from the inputs to the leaf ACS units of the LogLikelihood ratio processor-0 1250a before they are used.

[0143] The butterfly processors 1260 perform alpha computations, accumulating alpha values in the registers contained within constituent ACS units. The butterfly processors 1260 perform the forward trellis operation and normalisation as is usual during the forward trellis navigation.

[0144] The dummy betas calculated by the LogLikelihood ratio processor-0 1250a are presented to the butterfly processors 1260 at the start of the beta calculation phase.

[0145] During calculation of the betas, both LogLikelihood processors 1250a and 1250b are used in conjunction with the butterfly processors 1260. Each of the LogLikelihood processors 1250a, 1250b accepts alphas from the path metric store 1280, betas resulting from the previous clock cycle and extrinsic information 1242 produced from the Intermediate Decoding Memory and Processor 1240 to create a LogLikelihood result for a "1" and "0", respectively. The LogLikelihood calculations can span multiple rows since they are determining the maximum result over all the states.

[0146] Beta computations work in the reverse direction through the input symbol history window, compared to the alphas, and use gammas used in the alpha calculations. The beta computations use the same trellis branch metric assignments that were used for the alpha calculations.

[0147] When the whole block of input history has been processed and the resultant outputs have been fed into the interleaver 1520, the process is able to commence for the second half of the turbo decoder operation. The interleaver operation during first decoder operation is read sequentially and written sequentially. During second decoder operation, the interleaver is read from and written to, albeit using the random address sequence as determined by the scrambler address output. During second decoder operation, the read and write addresses are the same. The interleaver operation after the first decoder writes in sequentially and reads out randomly, as per the predefined spreading sequence which is used to give the first and second decoders their statistical independence. The interleaver operation for the second decoder writes randomly, as per the spreading sequence, and reads sequentially.

[0148] It is to be noted that because the encoders used for turbo encoding do not have to be the same, the decoding rates and constraints of the second decoder need not necessarily be the same as those for the first decoder. This may require that the configuration of the turbo decoder be changed between block processing operations. If this is the case, it is easily dealt with by manipulating the contents of the configuration registers.

[0149] Each block of input symbol history requires several complete turbo iterations in order to be decoded to within an acceptable bit error rate. The number of iterations required is configurable to ensure that the required bit error rate is achieved.

[0150] A benefit of the architecture in question is that it only requires two phases to complete one turbo decode iteration. This provides flexibility in the use of the architecture and allows the number of decoder rows used to be traded

for the number of iterations required. For example, a turbo decoder that does four iterations may be implemented using two decoder rows requiring two iteration times.

[0151] LogMAP computation is performed using a sliding window algorithm. The sliding window algorithm is implemented in 2 phases. In a single decoder this results in increased latency: 2 passes over each window as shown in the configuration (with only a single decoder being used) in Fig. 20A. The first pass computes the dummy beta values and the forward alpha values in parallel and stores the forward alpha values in the alpha memory (NOTE: this memory is the same memory as used in the Viterbi algorithm for path metric storage). The second pass reads the alpha values and computes the beta values according to the LogMAP algorithm and outputs LogLikelihood ratios (LLR).

[0152] When multiple decoders are used, the computation of the two phases can be overlapped and the decoder can process a single block with reduced latency. Multiple decoders can operate separately on different data streams or they can co-operate to increase the decoding speed of a single stream, as shown in the configuration of Fig. 20B. The implementation shown in Fig. 3 can process 4 independent streams or 2 streams with increased speed (and reduced latency) or I stream with further increased speed (and minimal latency).

[0153] Table 1 demonstrates the flexibility of the unified decoder to support multiple encoded streams simultaneously. For example, a decoder with 4 decoder rows can process up to 4 data streams at the same time. Furthermore, the decoder rows can operate together to decode fewer streams at higher throughput. This is useful for minimizing the latency of voice decoding. Table I demonstrates the flexibility of this approach and the appropriate decoding speed-up obtained in each case. (Again - this list is by no-means complete - more decoder rows can be connected together to achieve even greater flexibility.)

Table 1:

Example decoding configurations of multi-bank interconnected decoders.		
Scenario	Decoder Configuration	Decoding Speed-Up (over convolutional/ turbo on 1 decoder)
1 convolutional	4 decoders per stream	4X (conv)
2 convolutional	2 decoders per stream	2X (conv), 2X (conv)
3 convolutional	1 decoder for 2 streams, 2 decoders for 1 stream	1X (conv), 1X (conv), 2X (conv)
4 convolutional	1 decoder per stream	1X, 1X, 1X, 1X
1 turbo	2 decoders per stream	2X (turbo)
2 turbo	1 decoder per stream	1X, 1X
4 turbo	1 decoder per stream	1X, 1X, 1X, 1X
1 convolutional & 1 turbo	1 decoder conv, 2 decoders turbo	1X (conv), 2X (turbo)

[0154] To demonstrate how 2 or 4 decoders can co-operate to decode fewer data streams at a higher speed, Fig. 21 shows the interconnections between two decoders. The boxes marked "M" are multiplexers that enable some of the path metrics from adjacent decoders to be swapped before writing to the path metric memories. In this manner, the decoders can operate as a single decoder. Furthermore, Fig. 22 shows how 4 decoders can be interconnected to function as either a single decoder, two separate decoders, or 4 separate decoders.

[0155] To demonstrate the multi-standard nature of the unified decoder, the decoder can support any combination of the standards shown in Table 2. (This list is by no means complete - but is included to demonstrate the flexible (and therefore useful) nature of this unified decoder).

Table 2:

Example of standards supported by unified decoder.		
Standard	Code Rate	Constraint Length
GSM - full-rate voice	½	5
GSM - half-rate voice	1/3	7
GSM - data full-rate (9.6 Kbps)	½	5
GSM - data full rate (4.8 Kbps)	1/3	5

Table 2: (continued)

Example of standards supported by unified decoder.		
Standard	Code Rate	Constraint Length
GSM - data full rate (2.4 Kbps)	1/6	5
GPRS - CS-1	½	5
EDGE - MCS(1-9)	1/3	7
GSM-AMR TCH/AFS6.7	¼	7
GSM-AMR TCH/AFS5.15	1/5	7
UMTS Voice (slotted)	½	9
UMTS Voice (normal)	1/3	9
UMTS Data (Turbo)	½	4
UMTS Data (Turbo)	1/3	4
CDMA 2000 Voice	½	9
CDMA 2000 Voice	¼	9
CDMA 2000 Data (Turbo)	¼	4

[0156] The unified decoder 900 implements the decoding required for convolutional encoded and turbo encoded data streams and can support multiple data streams and multiple voice streams simultaneously. When decoding Turbo-encoded data streams, this decoder implements an iterative Turbo decoder using either the MAX-LOG MAP or the LOG-MAP soft-output MAP algorithms. The decoder maximizes the re-use of its components to enable the efficient implementation of both convolutional and turbo decoding systems.

[0157] The decoder can be dynamically partitioned, as required, to decode voice streams for different standards. The decoder can process streams with different coding rates (rate ½, rate 1/3, rate ¼, etc.). It can also process streams encoded with different constraint lengths. As such, the unified decoder architecture is capable of supporting each of the mobile wireless standards currently defined: first, second and third generation for both voice and data.

[0158] The unified decoder architecture of the preferred embodiment encapsulates the functionality of non-systematic (feed forward) encoders and systematic encoders (feed backward) in a single architecture. Fig. 23A shows mixing of polynomials 3240 and state bits 3250 to produce a single code bit 3225_0 of a code word 3225. Polynomials 3240 are presented to corresponding AND gates 3260, which also receive states 3250 as inputs. Each of the AND gates 3260 produces an output to a corresponding XOR gate 3270. Each XOR gate 3270 also receives a TRANSITION INPUT 3280 and produces an output 3225_0 of the M-bit non-systematic encoder 3230.

[0159] Fig. 23B shows a whole encoder 3200 for a code word 3225. Polynomials 3240 are presented to corresponding M-bit non-systematic encoders 3230. An input bit 3220 is presented to an XOR gate 3275. A RSC_ENABLE signal is presented to an AND gate 3280, the output of which is the second input of the XOR gate 3275. The AND gate 3280 also receives as an input the output of the encoders 3230. The XOR gate 3275 presents an output to an M-bit shift register 3210 and to each of the encoders 3230. The M-Bit Shift Register 3210 also receives a clock signal 3285 and a reset signal 3290 and holds a state of the encoder 3200 at a time T. The state value is used in conjunction with each particular polynomial 3240 (as specified by a particular code) to produce a non-systematic code bit. The output 3250 of the register 3210 is broadcast to each of the encoders 3230. The outputs 3225_0...3225_R of the encoder 3230 are collated to form the CODE_WORD 3225.

[0160] By enabling the RSC_ENABLE 3215, the encoder 3200 becomes a recursive, systematic (RS) encoder. In a recursive, systematic code, the input bit 3220 forms the systematic bit of a code word 3225. The generated bits of each M-Bit Encoder 3230 form the remainder of the RS code word 3225.

[0161] In the case of a non-systematic encoder the CODE_WORD 3225 would contain R bits (where R = the rate of the code). When the RSC_ENABLE 3215 is active, the CODE_WORD 3225 is typically 1-bit wide. The output CODE_WORD 3225 (in this case 1-bit wide) and the INPUT_BIT 3220 form the RS code word.

[0162] It is apparent from the above that the embodiment(s) of the invention are applicable to the decoding of multiple wireless transmission standards using a unified, scalable architecture.

[0163] The foregoing describes only one embodiment/some embodiments of the present invention, and modifications and/or changes can be made thereto without departing from the scope of the invention, the embodiment(s) being illustrative and not restrictive.

Claims

1. A reconfigurable architecture for decoding data communications signals transmitted according to one of a plurality of coding schemes, said coding schemes comprising convolutional and turbo codes, said architecture comprising:

   a trellis processing arrangement for receiving an input signal derived from said transmitted signals and new path metrics for determining intermediate decoded results using said new path metrics;
   an intermediate store for receiving modified decoded results and for providing a decoded output; and
   control processor means coupled to said trellis processing arrangement and operable to configure said architecture for one of convolutional or turbo decoding by (i) developing said new path metrics using generated path metrics output from the trellis processing arrangement, (ii) determining said modified decoded results from said intermediate decoded results, and (iii) for determining said decoded output from a selected one of said modified decoded results.

2. A reconfigurable architecture as claimed in claim 1, wherein when said control processor means configures said architecture for convolutional decoding using a Viterbi algorithm, said new path metrics comprise said generated path metrics, said intermediate decoding results comprise decision bits from said trellis processing arrangement, and said intermediate store and said control arrangement implement traceback processing of said intermediate decoded results to provide a convolutionally decoded output.

3. A reconfigurable architecture as claimed in claim 1, wherein when said control processor means configures said architecture for convolutional decoding using a LogMAP algorithm, said trellis processing arrangement implements LogMAP trellis processing and said intermediate store forms part of a store within said trellis processing arrangement.

4. A reconfigurable architecture as claimed in claim 1, wherein when said control processor means configures said architecture for turbo decoding, said intermediate decoding results comprise extrinsic information that is modified by said control processor means using interleaving, and said trellis processing arrangement implementing LogMAP trellis processing to produce new extrinsic information supplied to said intermediate store, to provide a turbo decoded output.

5. A reconfigurable architecture as claimed in any of claims 2,3 or 4 wherein said trellis processing arrangement comprises:

   butterfly processor means configured for receiving said input symbol and for generating intermediate decoded results and generated path metrics;
   a first path metric store for providing old path metrics to said butterfly processor means; and
   a second path metric store for receiving new path metrics and buffering said new path metrics to said first path metric store.

6. A reconfigurable architecture as claimed in claim 5 wherein said intermediate store forms part of said first path metric store.

7. A reconfigurable architecture as claimed in claim 5 wherein said first and second path metric stores are formed by a double buffered memory arrangement.

8. A reconfigurable architecture as claimed in claim 5 wherein said first and second stores comprise registers.

9. A reconfigurable architecture as claimed in claim 5 wherein said second path metric store is formed using a plurality of hold registers, said architecture further comprising a multiplexer arrangement interconnecting said hold registers and said first path metric store, said multiplexer arrangement being configurable by said control processor means to update said first path metric store with a specifically ordered sequence of said new path metrics supplied to said hold registers in an originating sequence.

10. A telecommunications decoding device comprising:

    a parallel arrangement of decoding processors and at least one store each arranged in a process loop conveying decoding process values received by and generated from said decoding processors;

wherein said decoding processors receive coded data and present decoded results, and a flow of said decoding process values about said process loop is controlled to alter a decoding function performed by said device from one decoding scheme to at least one other decoding scheme.

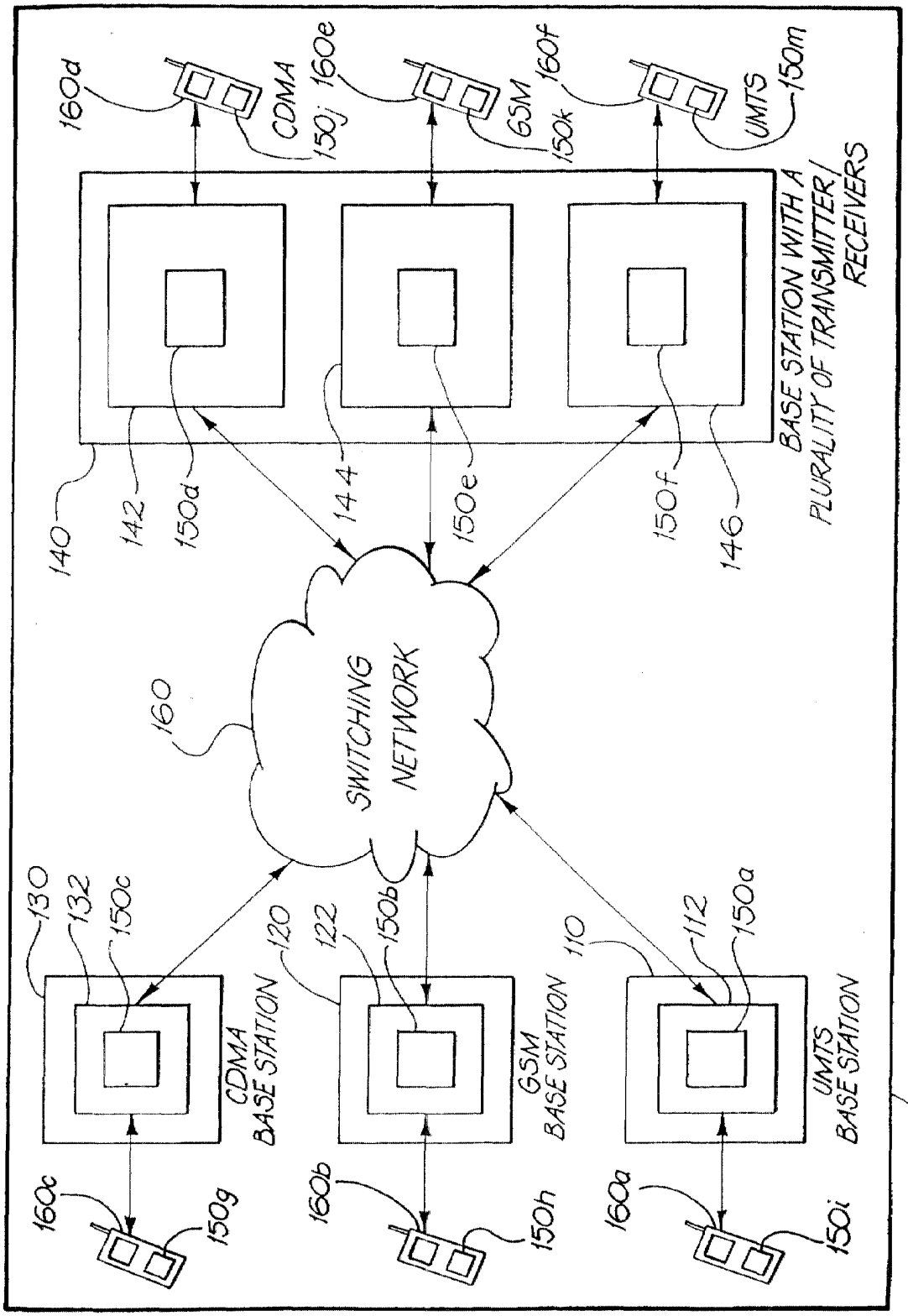

160d
CDMA
150j
160e
GSM
150k
160f
UMTS
150m
BASE STATION WITH A PLURALITY OF TRANSMITTER/ RECEIVERS
140
142
150a
144
150e
150f
146
160
SWITCHING NETWORK
130
132
150c
CDMA BASE STATION
120
122
150b
GSM BASE STATION
110
112
150a
UMTS BASE STATION
160c
150g
160b
150h
160a
150i
100

FIG. 1

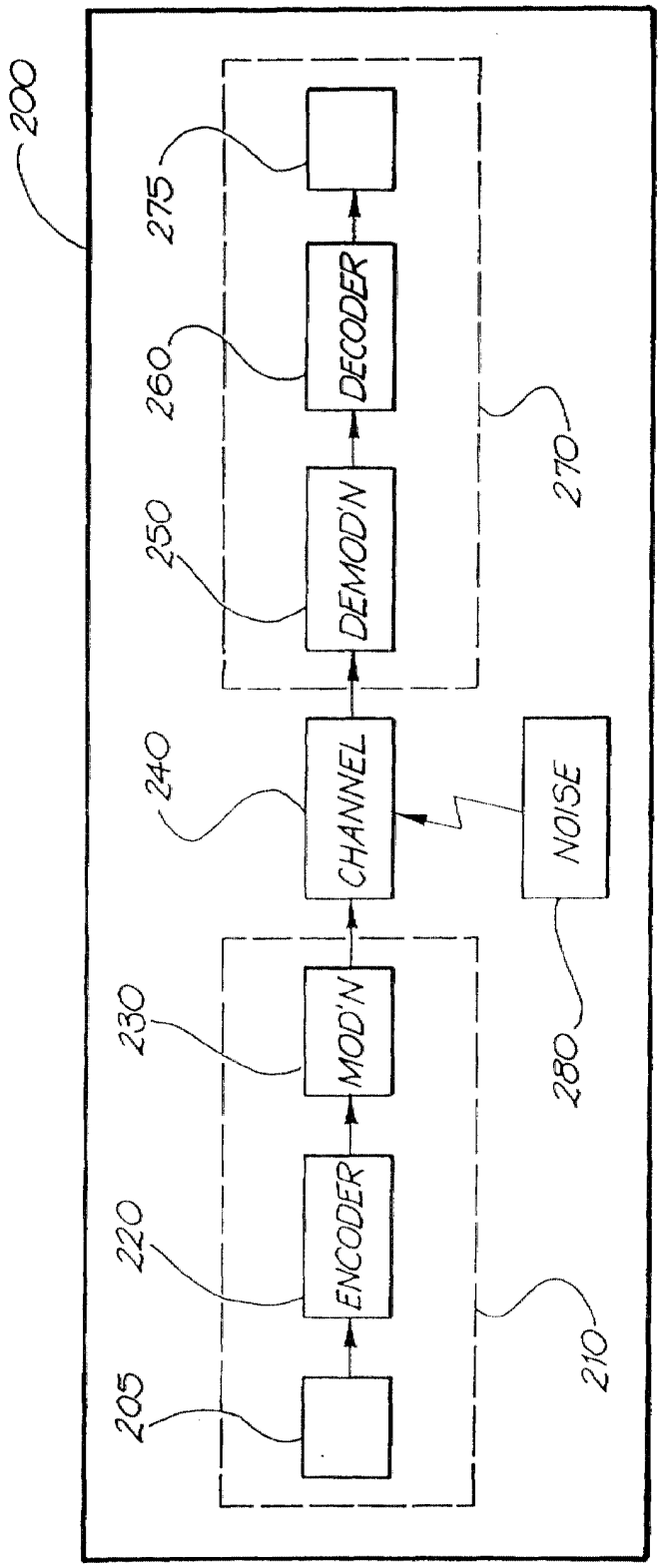


FIG. 2A
(PRIOR ART)

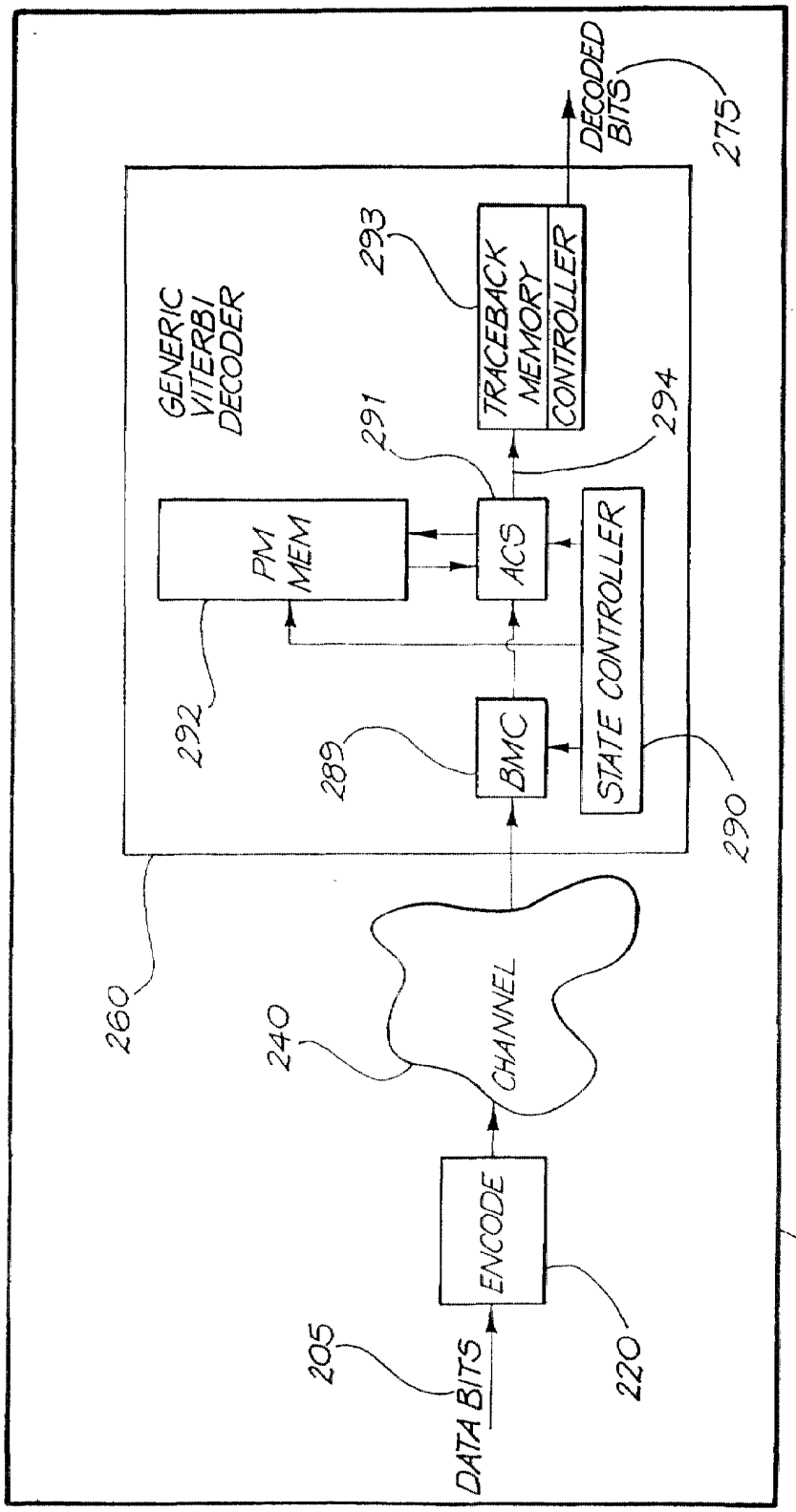


FIG. 2B
(PRIOR ART)

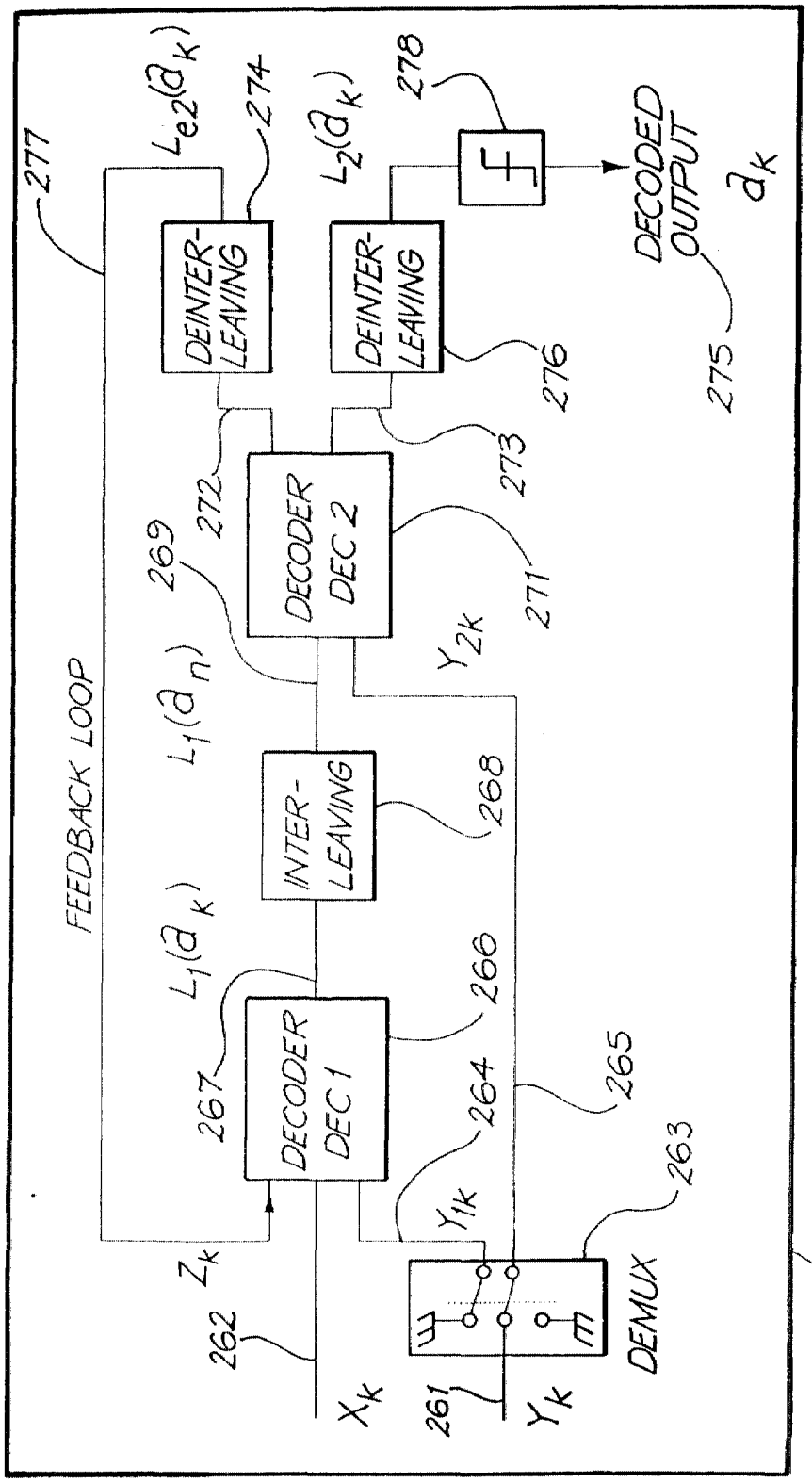


FIG. 2C
(PRIOR ART)

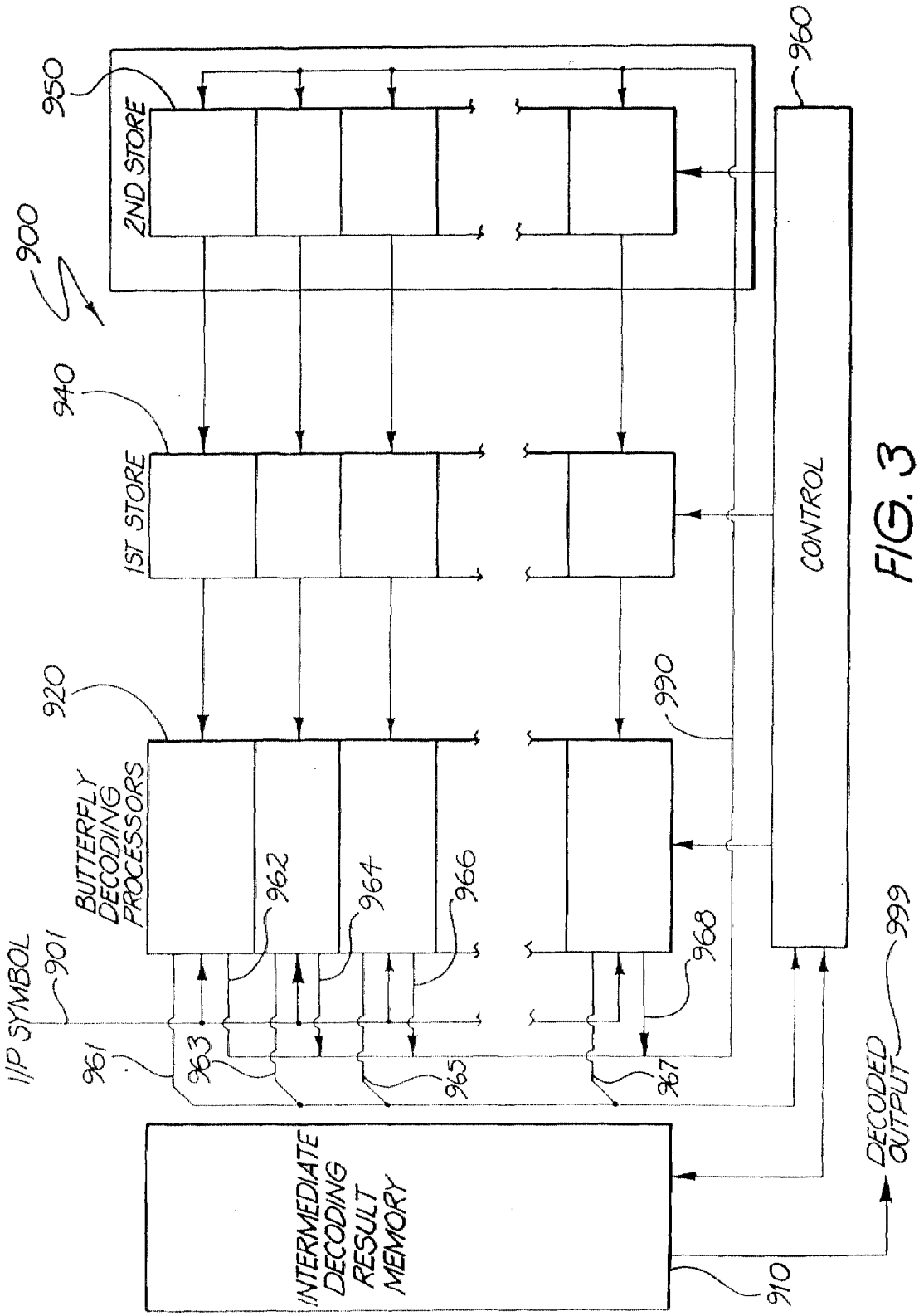

950
2ND STORE
900
940
1ST STORE
920
BUTTERFLY DECODING PROCESSORS
962
964
966
968
990
960
CONTROL
I/P SYMBOL
901
961
963
965
967
999
DECODED OUTPUT
INTERMEDIATE DECODING RESULT MEMORY
910

FIG. 3

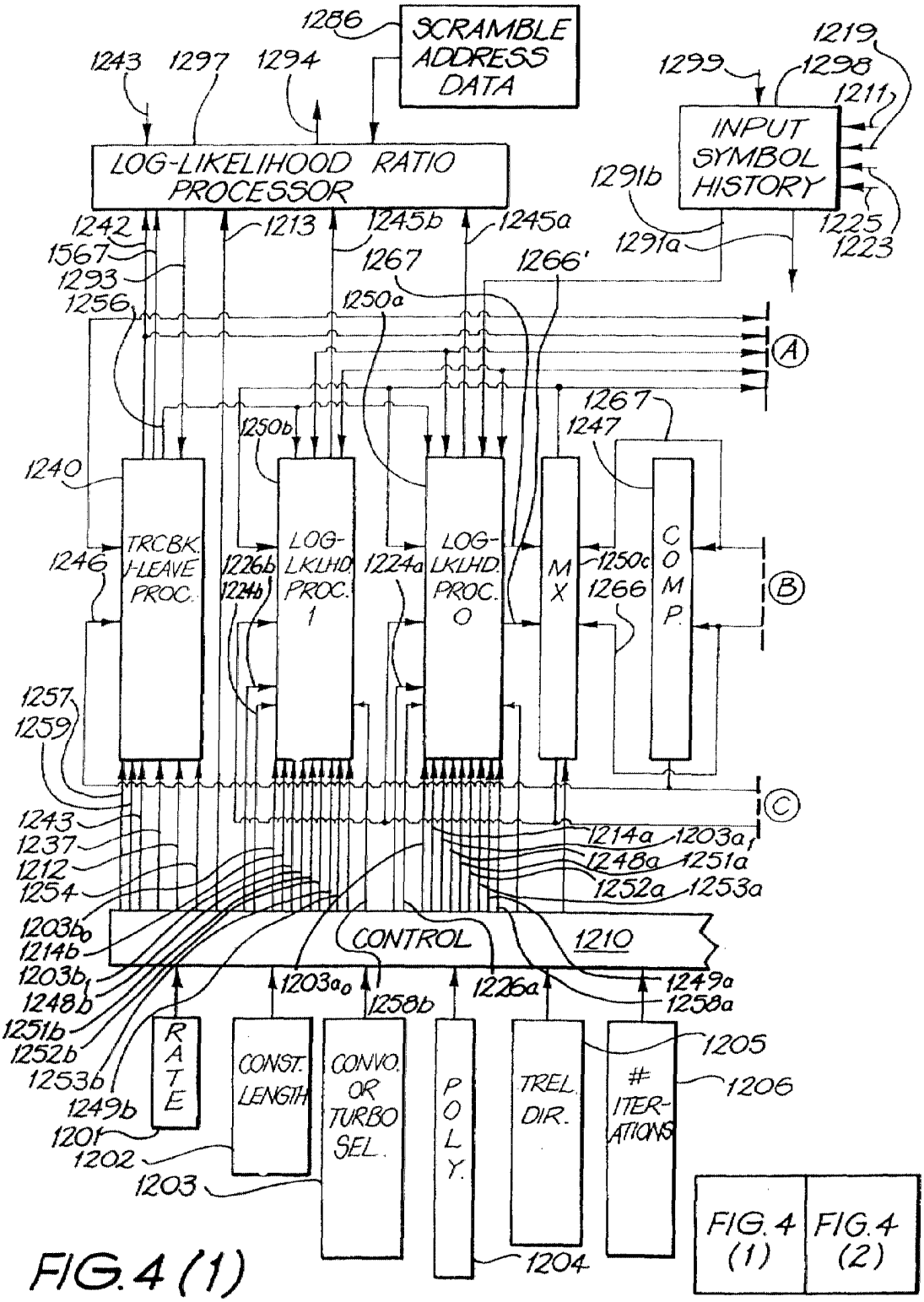


FIG. 4 (1)

FIG. 4 (1)	FIG. 4 (2)

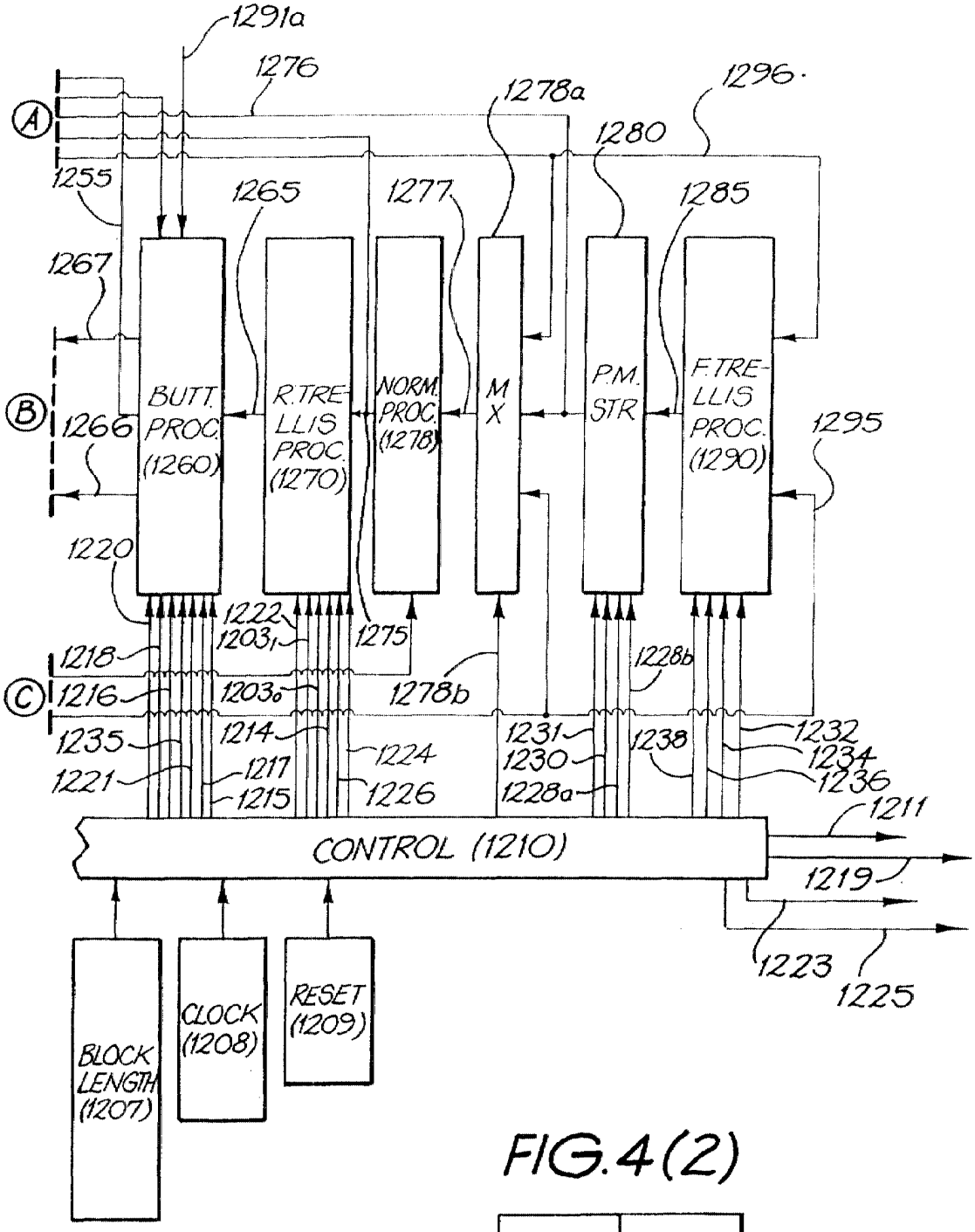


FIG. 4 (2)

FIG. 4 (1)	FIG. 4 (2)

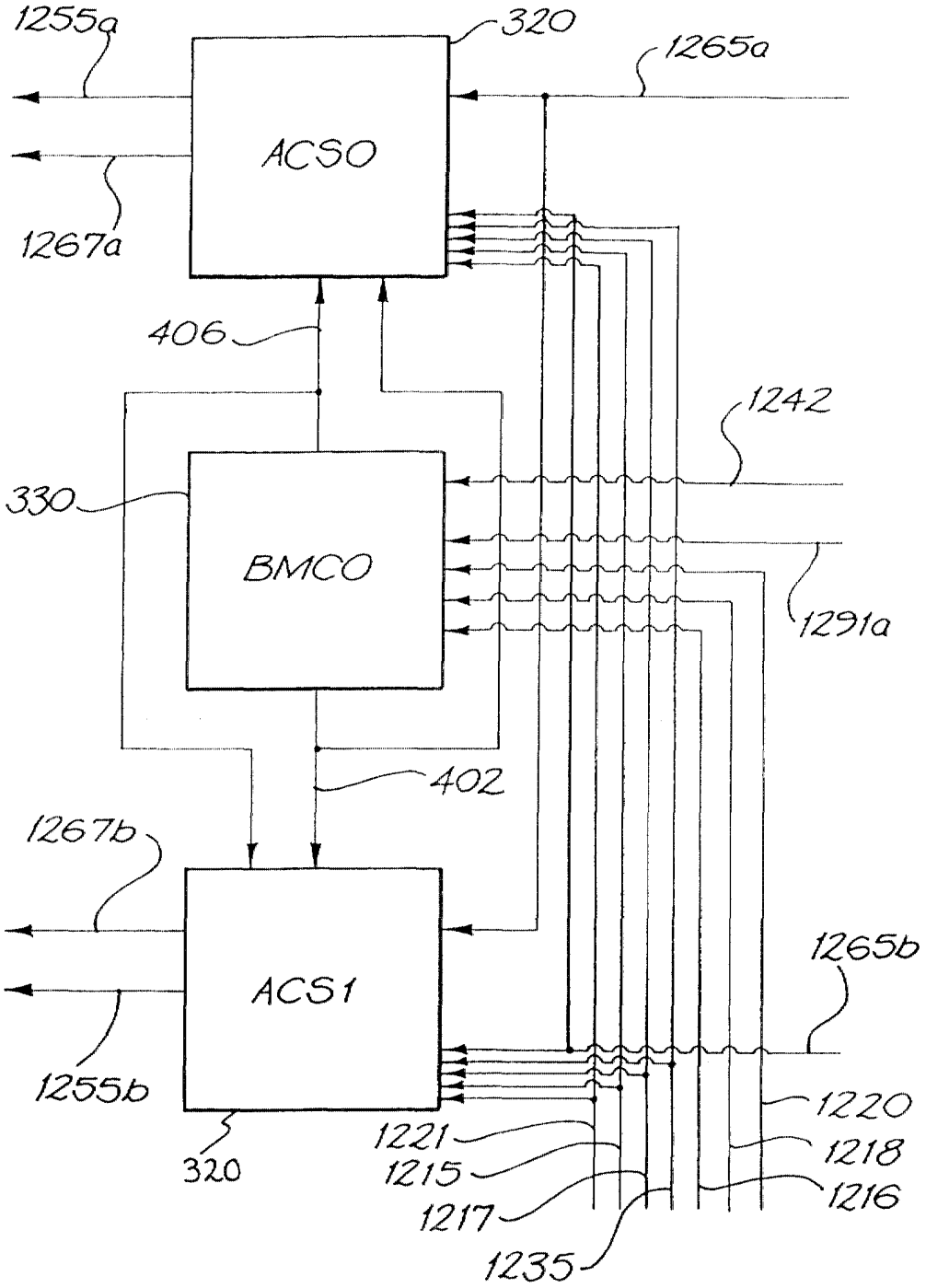

1255a
320
1265a
ACS0
1267a
406
1242
330
BMC0
1291a
402
1267b
1265b
ACS1
1255b
1220
1221
1218
1215
1216
1217
320
1235

FIG. 5A

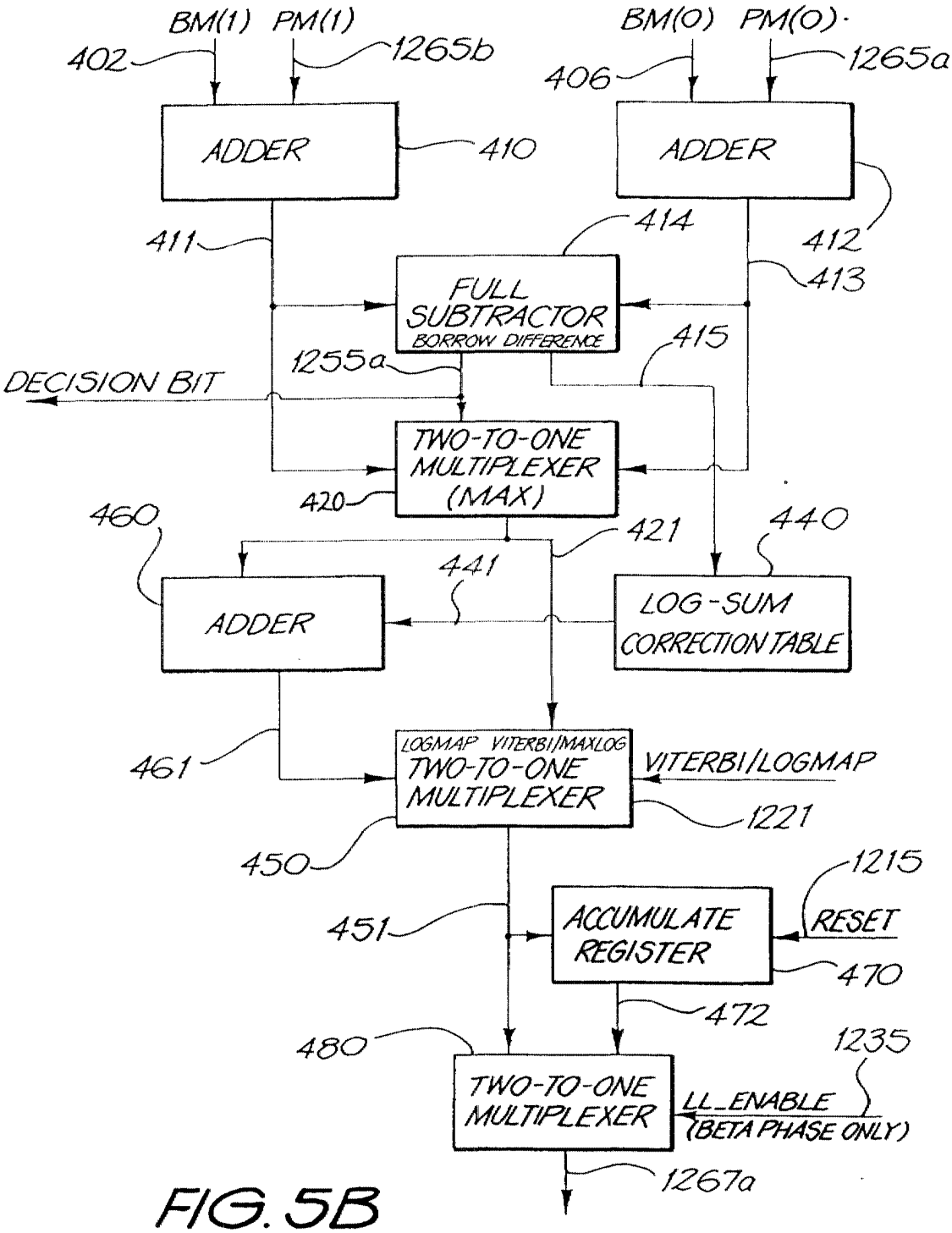

BM(1) PM(1)
402
1265b
BM(0) PM(0)
406
1265a
ADDER
410
ADDER
411
414
412
FULL SUBTRACTOR
BORROW DIFFERENCE
413
415
1255a
DECISION BIT
TWO-TO-ONE MULTIPLEXER (MAX)
420
460
421
440
441
ADDER
LOG-SUM CORRECTION TABLE
LOGMAP VITERBI/MAXLOG
TWO-TO-ONE MULTIPLEXER
VITERBI/LOGMAP
461
1221
450
1215
451
ACCUMULATE REGISTER
RESET
470
472
480
1235
TWO-TO-ONE MULTIPLEXER
LL_ENABLE
(BETA PHASE ONLY)
1267a

FIG. 5B

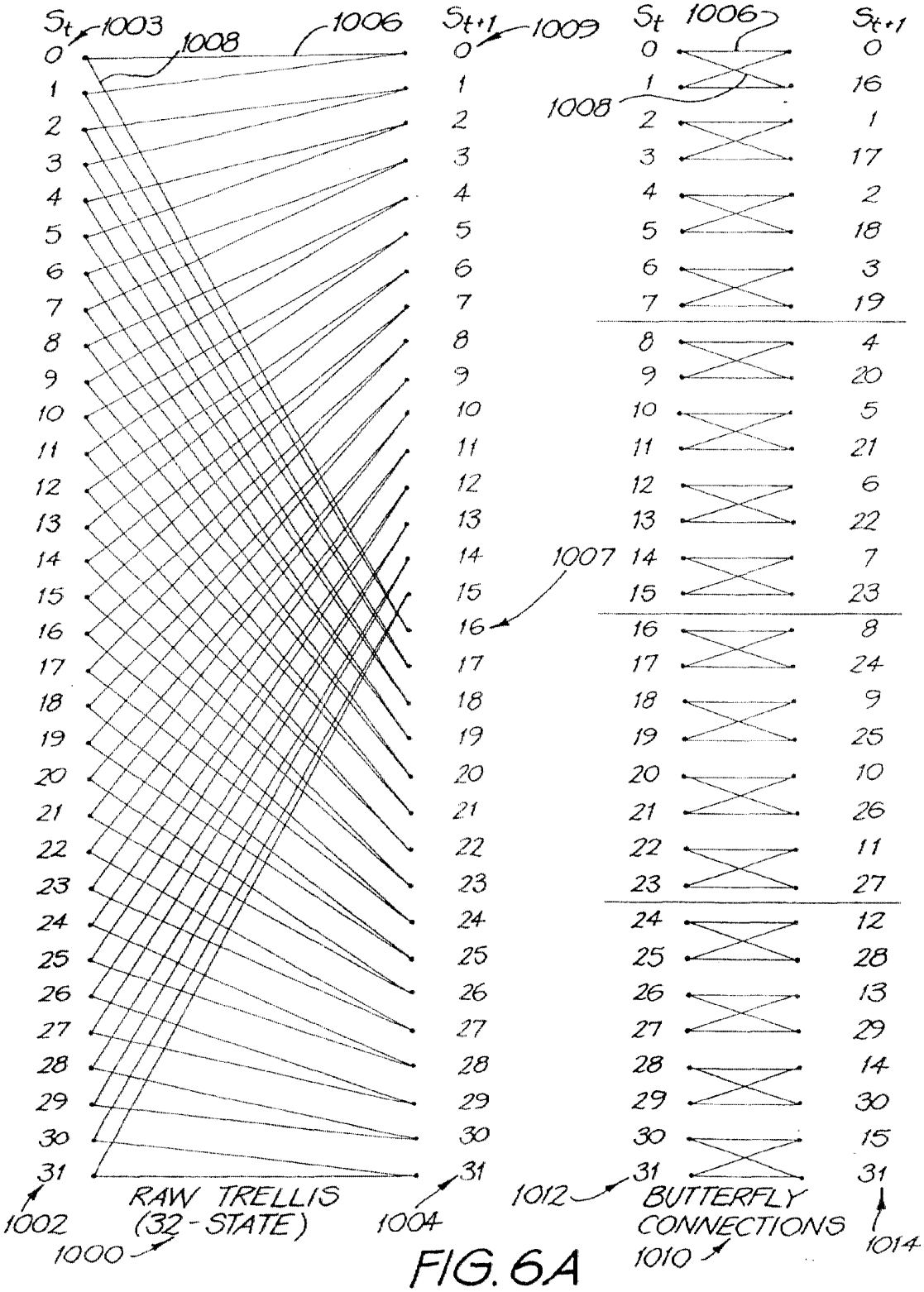

St
1003
1008
1006
St+1
1009
St
1006
St+1
1008
1007
1002
RAW TRELLIS
(32-STATE)
1000
1004
1012
BUTTERFLY
CONNECTIONS
1010
1014

FIG. 6A

RESULTANT PATH METRIC LOCATIONS			
0	4	8	12
16	20	24	28
1	5	9	13
17	21	25	29
2	6	10	14
18	22	26	30
3	7	11	15
19	23	27	31

1022 1024 1026 1028

FIG. 6B

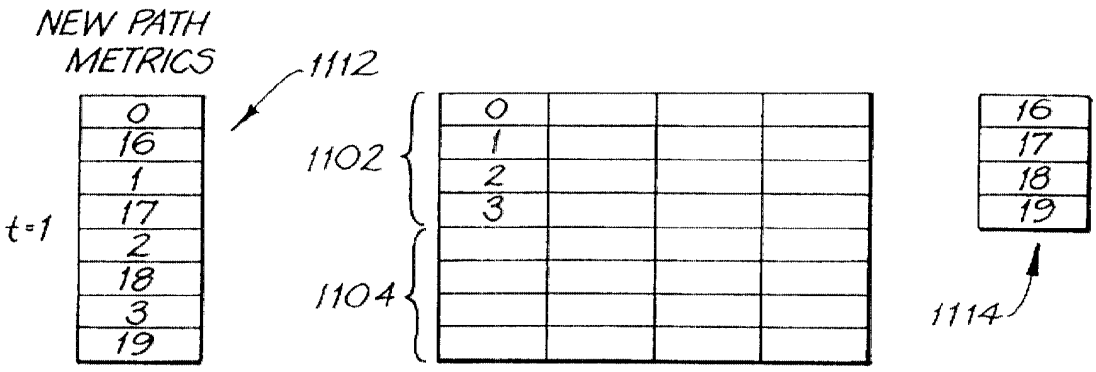


FIG. 7A

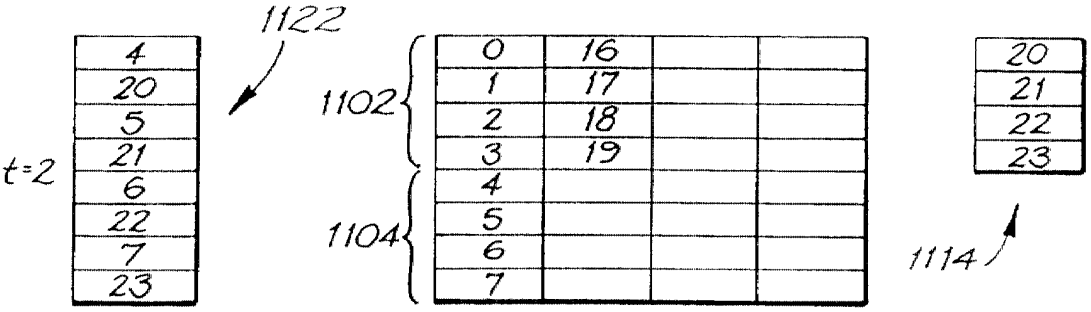


FIG. 7B

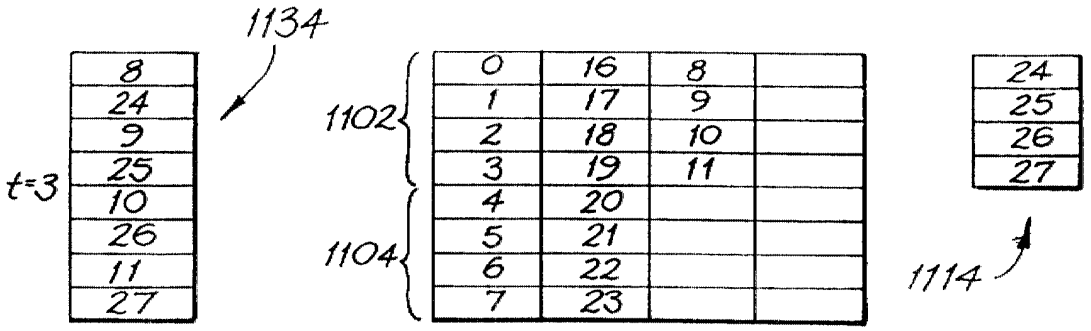


FIG. 7C

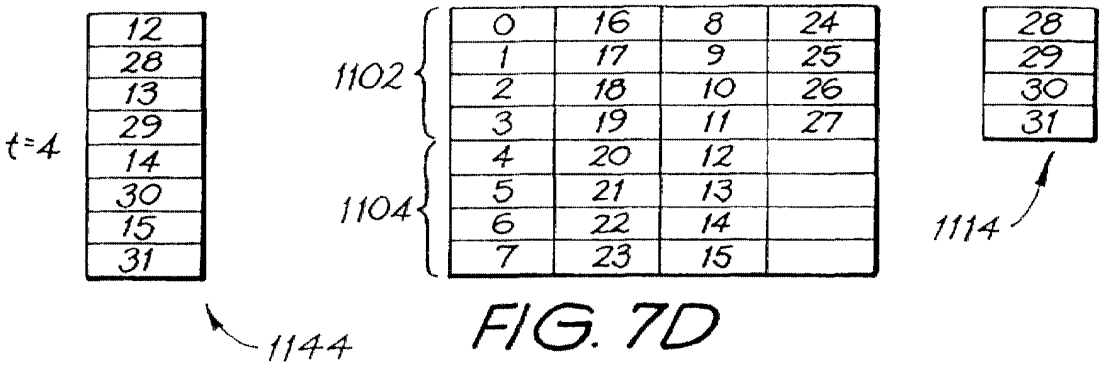


FIG. 7D

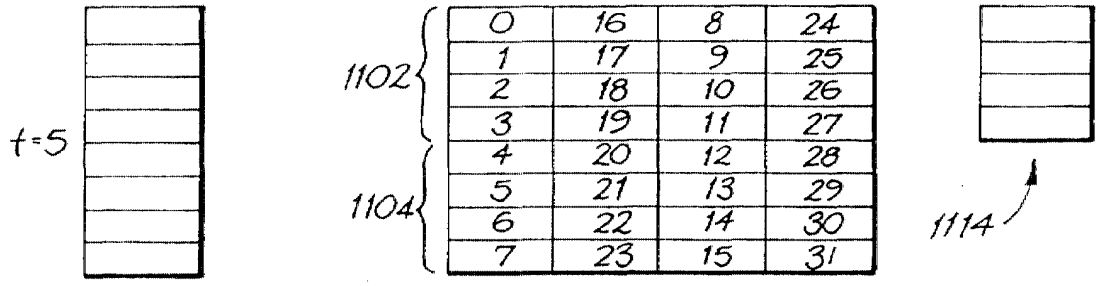


FIG. 7E

	PATH METRIC COLUMNS				ADDRESSING SEQUENCE OF COLUMNS
	3	2	0	1	
Sn	24-31	16-23	0-7	8-15	0,1,2,3
Sn+1	24-31	8-15	0-7	16-23	0,2,1,3
Sn+2	24-31	16-23	0-7	8-15	0,1,2,3

1160

1150

FIG. 7F

A	0 1 2 3	8 9 10 11	16 17 18 19	24 25 26 27
B	4 5 6 7	12 13 14 15	20 21 22 23	28 29 30 31

≡

A	C_{0A}	C_{1A}	C_{2A}	C_{3A}
B	C_{0B}	C_{1B}	C_{2B}	C_{3B}

FIG. 8A

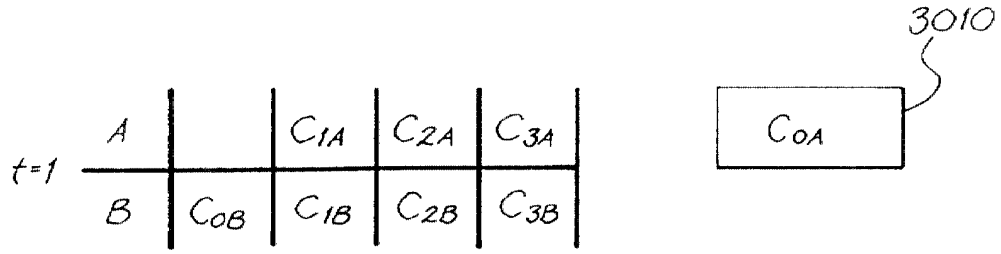


FIG. 8B

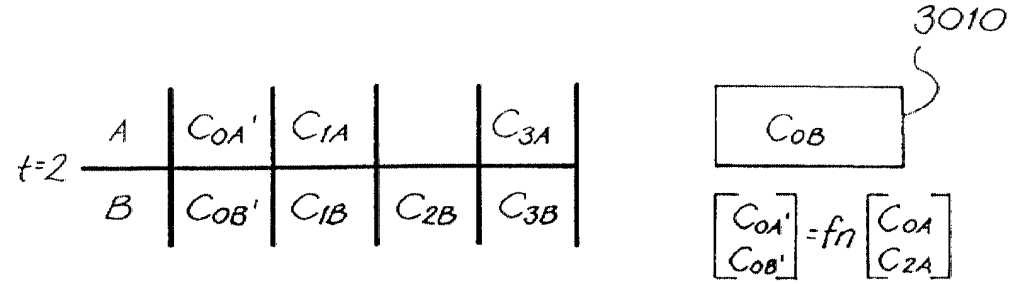


FIG. 8C

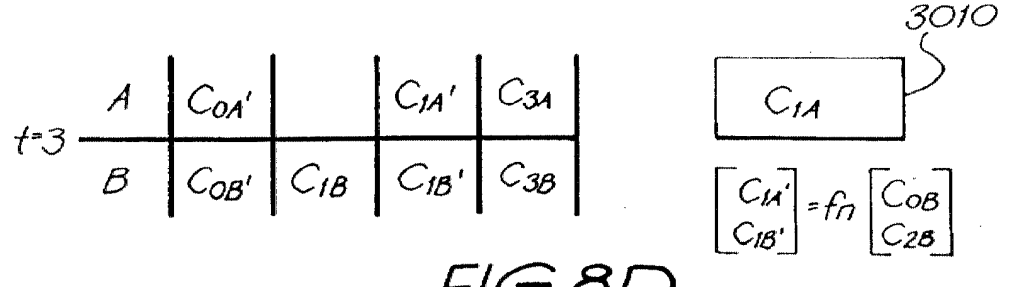


FIG. 8D

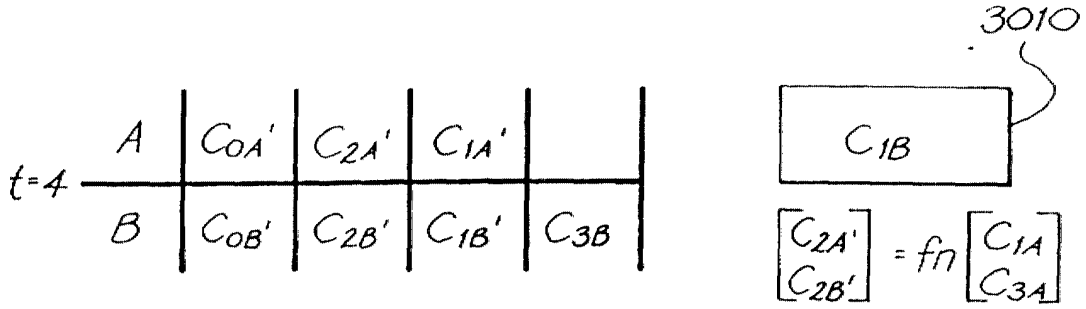

t=4
A
B
C0A'
C2A'
C1A'
C0B'
C2B'
C1B'
C3B
3010
C1B
[C2A' C2B'] = fn [C1A C3A]

FIG. 8E

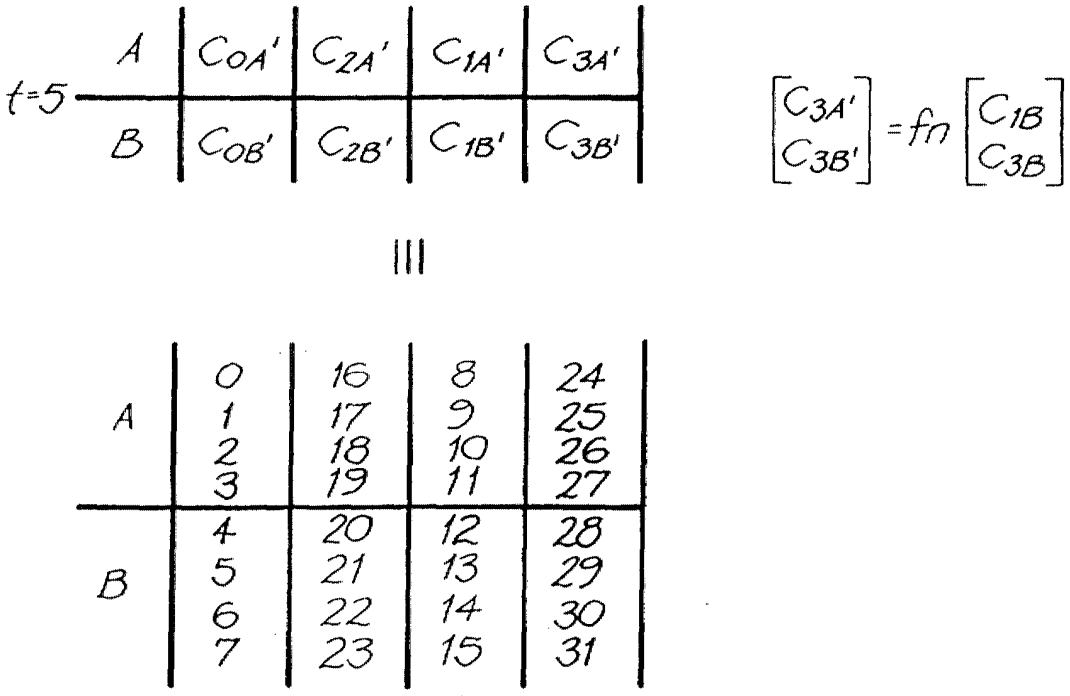

t=5
A
B
C0A'
C2A'
C1A'
C3A'
C0B'
C2B'
C1B'
C3B'
[C3A' C3B'] = fn [C1B C3B]
|||
0 16 8 24
1 17 9 25
2 18 10 26
3 19 11 27
4 20 12 28
5 21 13 29
6 22 14 30
7 23 15 31

FIG. 8F

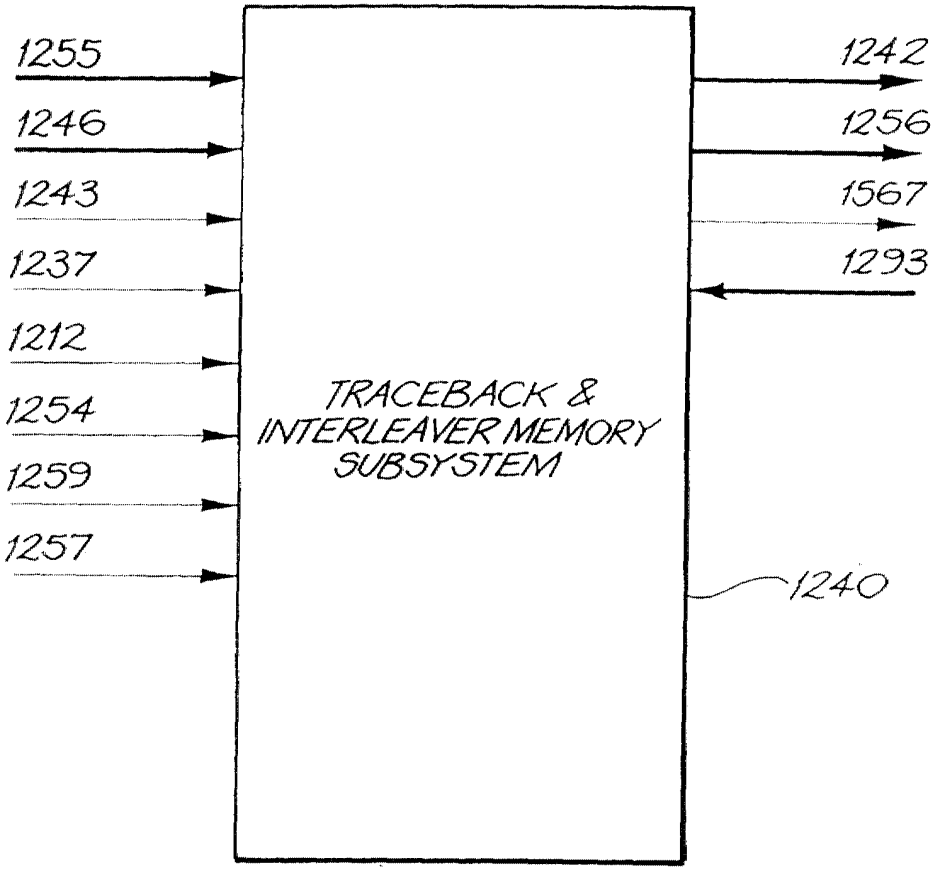

1255
1246
1243
1237
1212
1254
1259
1257
TRACEBACK &
INTERLEAVER MEMORY
SUBSYSTEM
1242
1256
1567
1293
1240

FIG. 9A

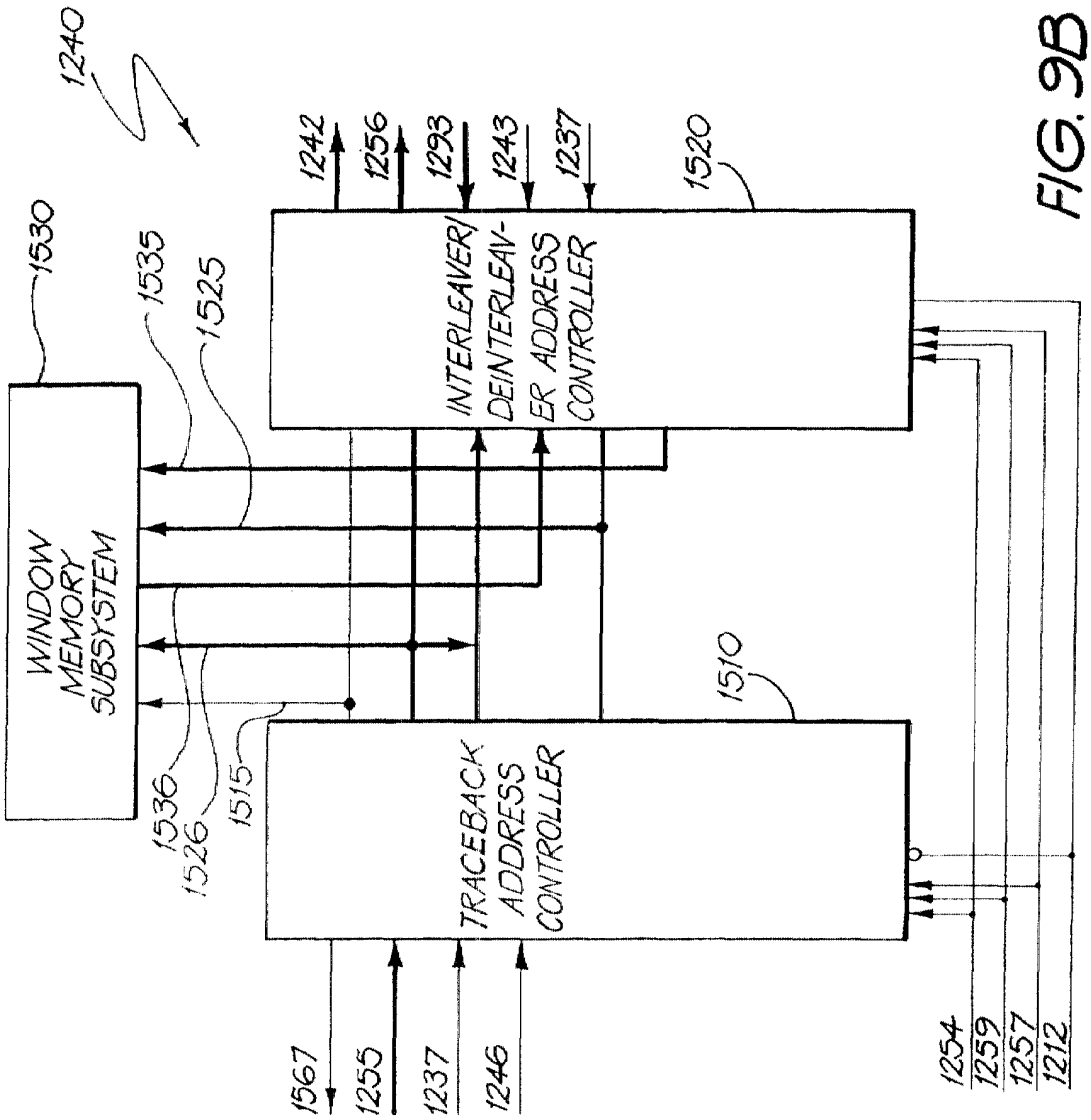

1240
1242
1256
1293
1243
1237
1520
1530
1535
1525
INTERLEAVER/
DEINTERLEAV-
ER ADDRESS
CONTROLLER
WINDOW
MEMORY
SUBSYSTEM
1510
1536
1526
1515
TRACEBACK
ADDRESS
CONTROLLER
1567
1255
1237
1246
1254
1259
1257
1212

FIG. 9B

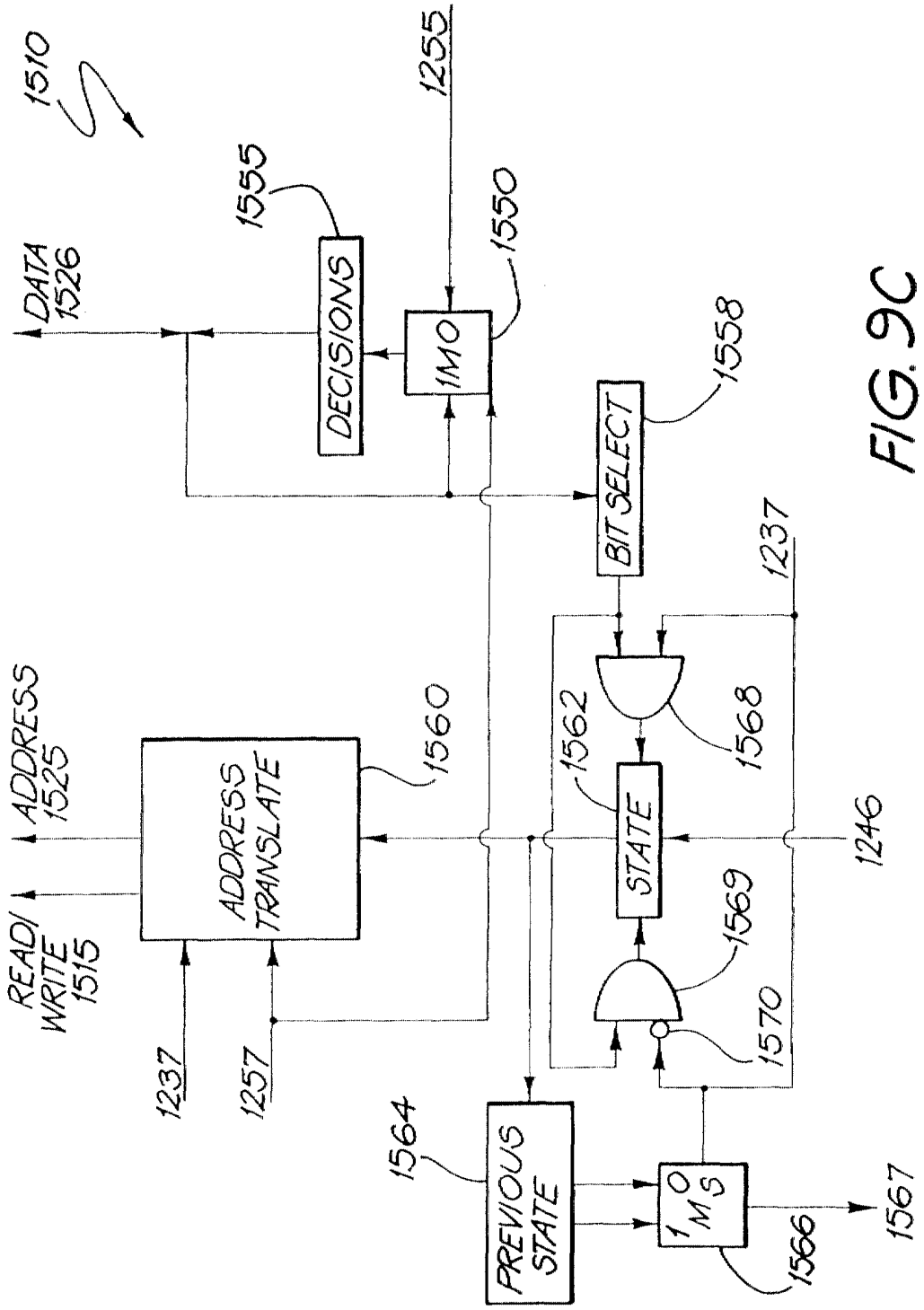

1510
1555
1255
DATA 1526
DECISIONS
IMO
1550
1558
BIT SELECT
1237
ADDRESS 1525
READ/ WRITE 1515
ADDRESS TRANSLATE
1560
1562
1568
STATE
1246
1569
1570
1237
1257
1564
PREVIOUS STATE
1 M S 0
1566
1567

FIG. 9C

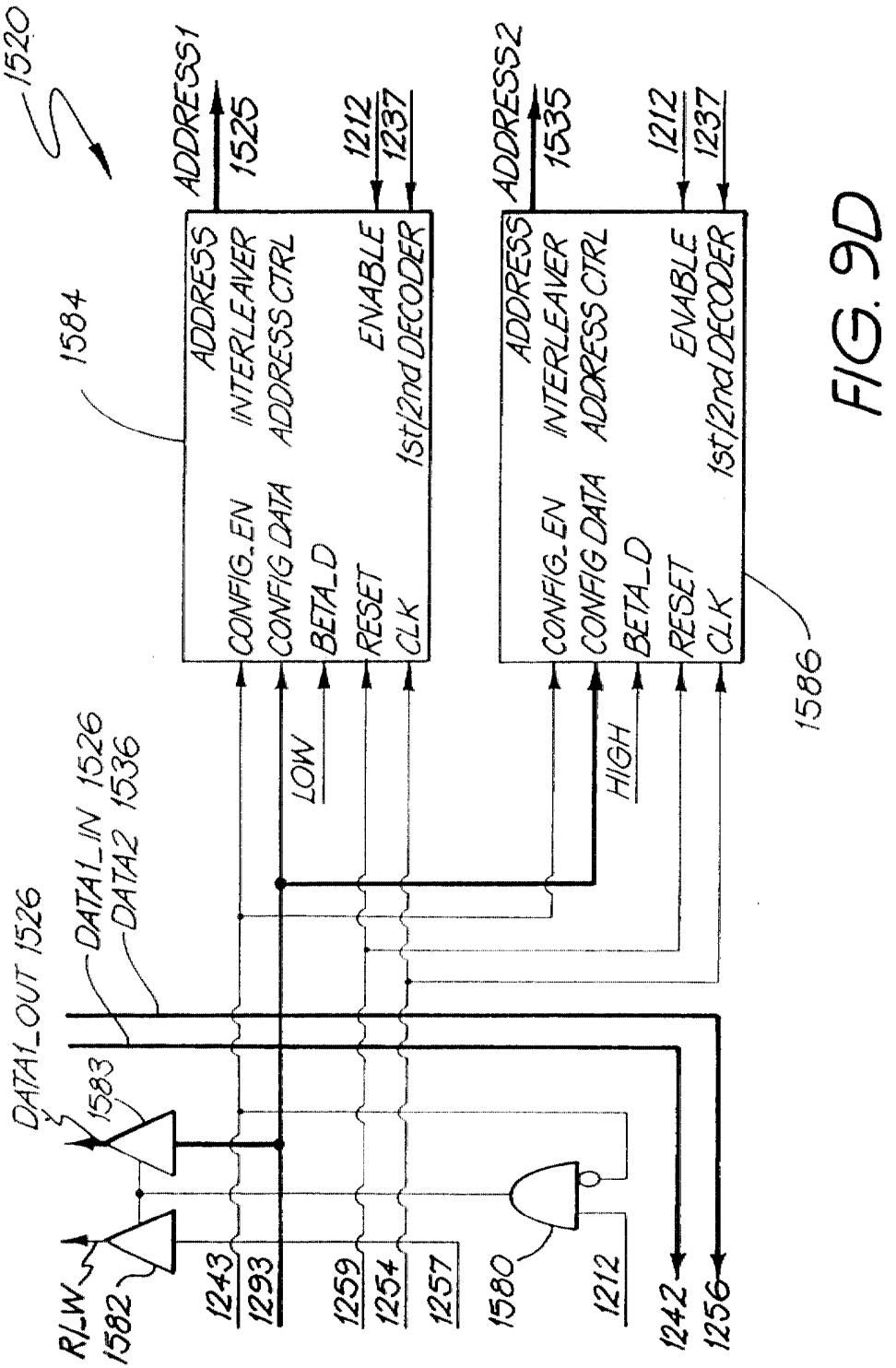

1520
ADDRESS1
1525
1212
1237
1584
ADDRESS INTERLEAVER
ADDRESS CTRL
ENABLE
1st/2nd DECODER
CONFIG_EN
CONFIG DATA
BETA_D
RESET
CLK
ADDRESS2
1535
1212
1237
ADDRESS INTERLEAVER
ADDRESS CTRL
ENABLE
1st/2nd DECODER
CONFIG_EN
CONFIG DATA
BETA_D
RESET
CLK
1586
DATA1_IN 1526
DATA2 1536
DATA1_OUT 1526
1583
R/W
1582
LOW
HIGH
1243
1293
1259
1254
1257
1580
1212
1242
1256

FIG. 9D

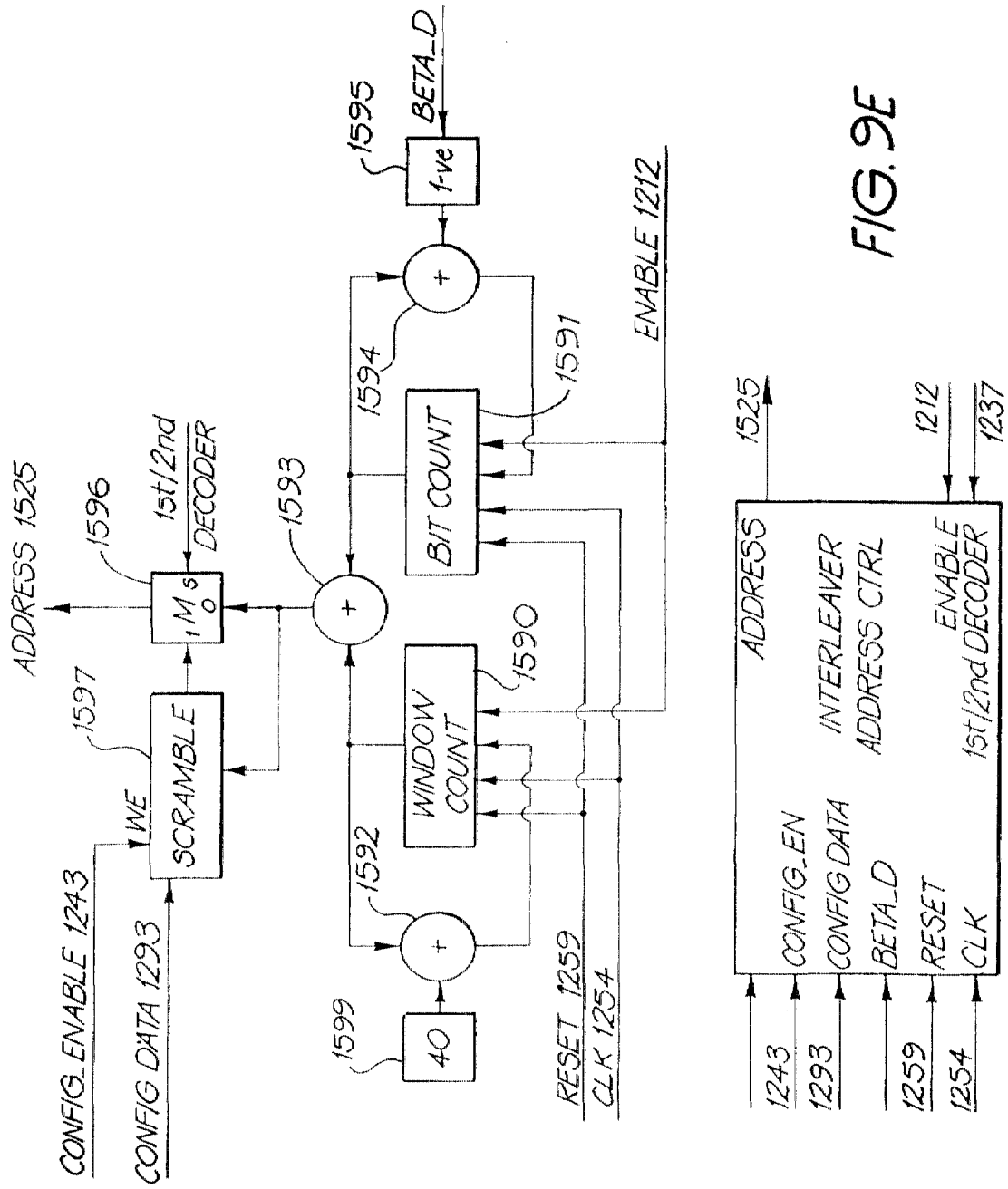

CONFIG_ENABLE 1243
CONFIG DATA 1293
WE
SCRAMBLE
1597
ADDRESS 1525
1596
1st/2nd DECODER
1 M S 0
1593
1599
40
1592
WINDOW COUNT
1590
BIT COUNT
1591
1594
1595
1-ve
BETA_D
ENABLE 1212
RESET 1259
CLK 1254
ADDRESS
1525
CONFIG_EN
CONFIG DATA
BETA_D
RESET
CLK
INTERLEAVER ADDRESS CTRL
ENABLE 1st/2nd DECODER
1212
1237
1243
1293
1259
1254

FIG. 9E

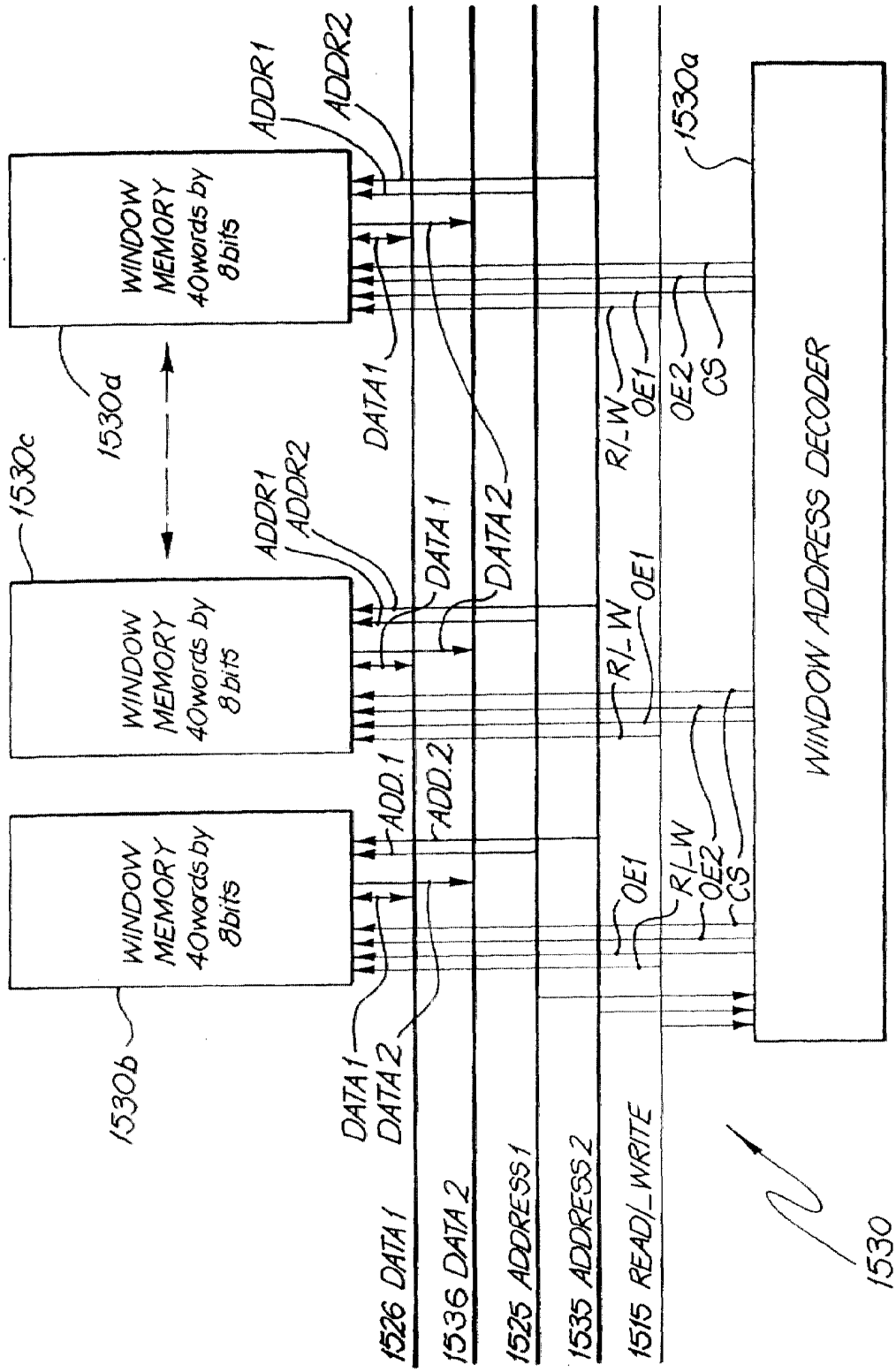

WINDOW MEMORY 40words by 8bits
1530d
1530c
1530b
ADDR1
ADDR2
DATA1
DATA2
ADD.1
ADD.2
R/_W
OE1
OE2
CS
1530a
WINDOW ADDRESS DECODER
1526 DATA 1
1536 DATA 2
1525 ADDRESS 1
1535 ADDRESS 2
1515 READ/_WRITE
1530

FIG. 9F

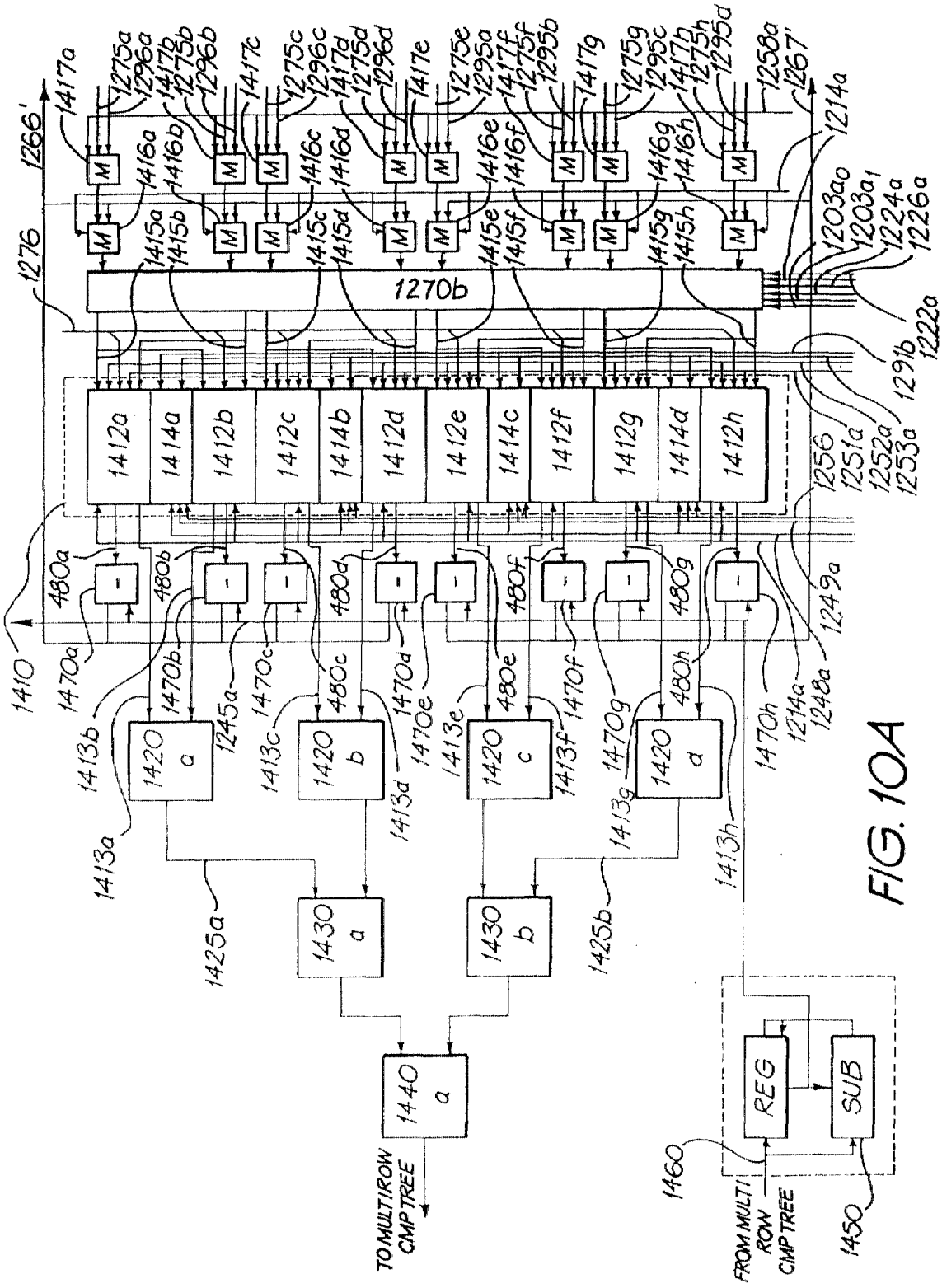

1270b
1412a
1414a
1412b
1412c
1414b
1412d
1412e
1414c
1412f
1412g
1414d
1412h
1420 a
1420 b
1420 c
1420 d
1430 a
1430 b
1440 a
REG
SUB
TO MULTI ROW CMP TREE
FROM MULTI ROW CMP TREE

FIG. 10A

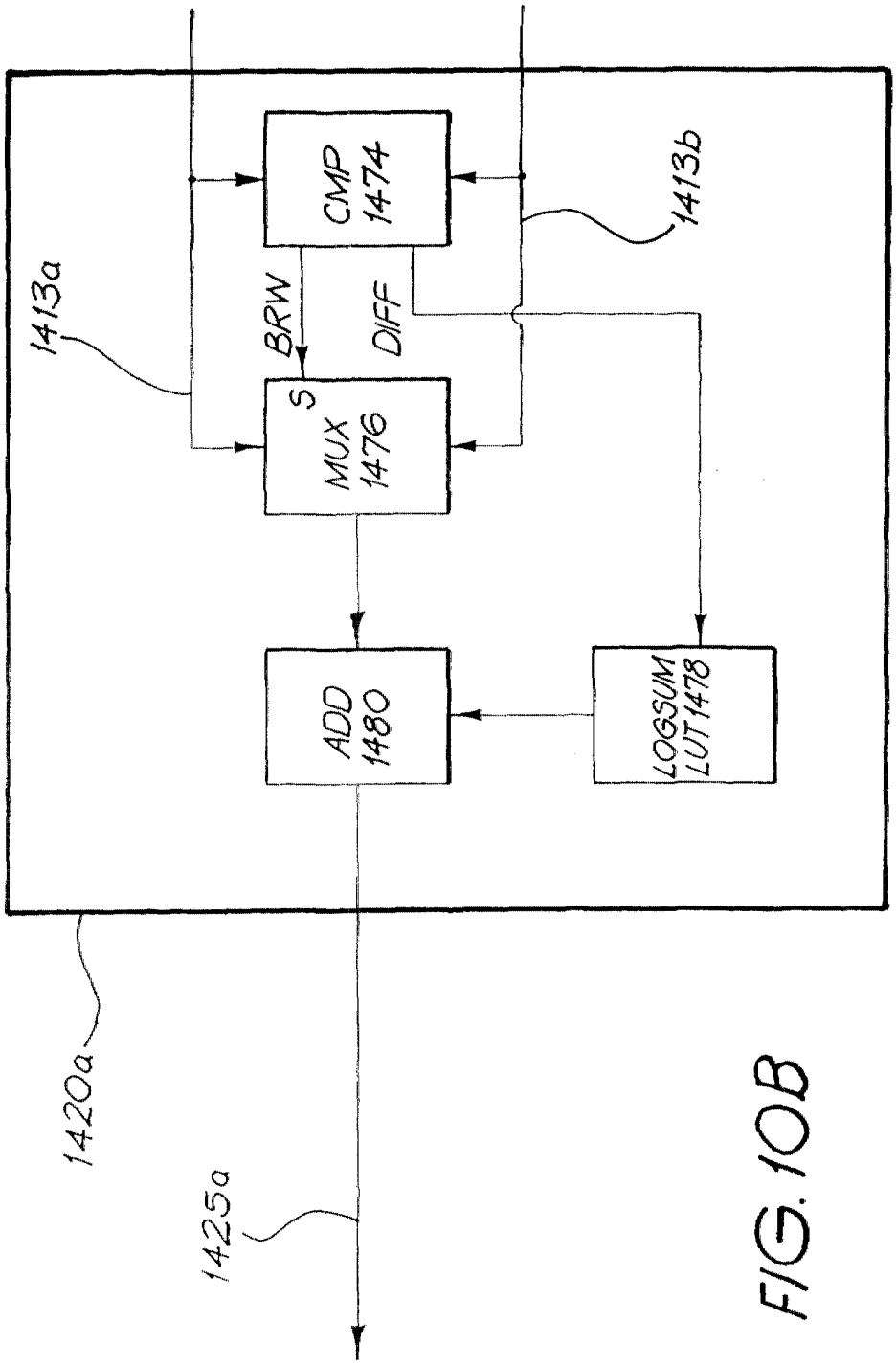

CMP
1474
1413b
1413a
BRW
DIFF
S
MUX
1476
ADD
1480
LOGSUM
LUT1478
1420a
1425a

FIG. 10B

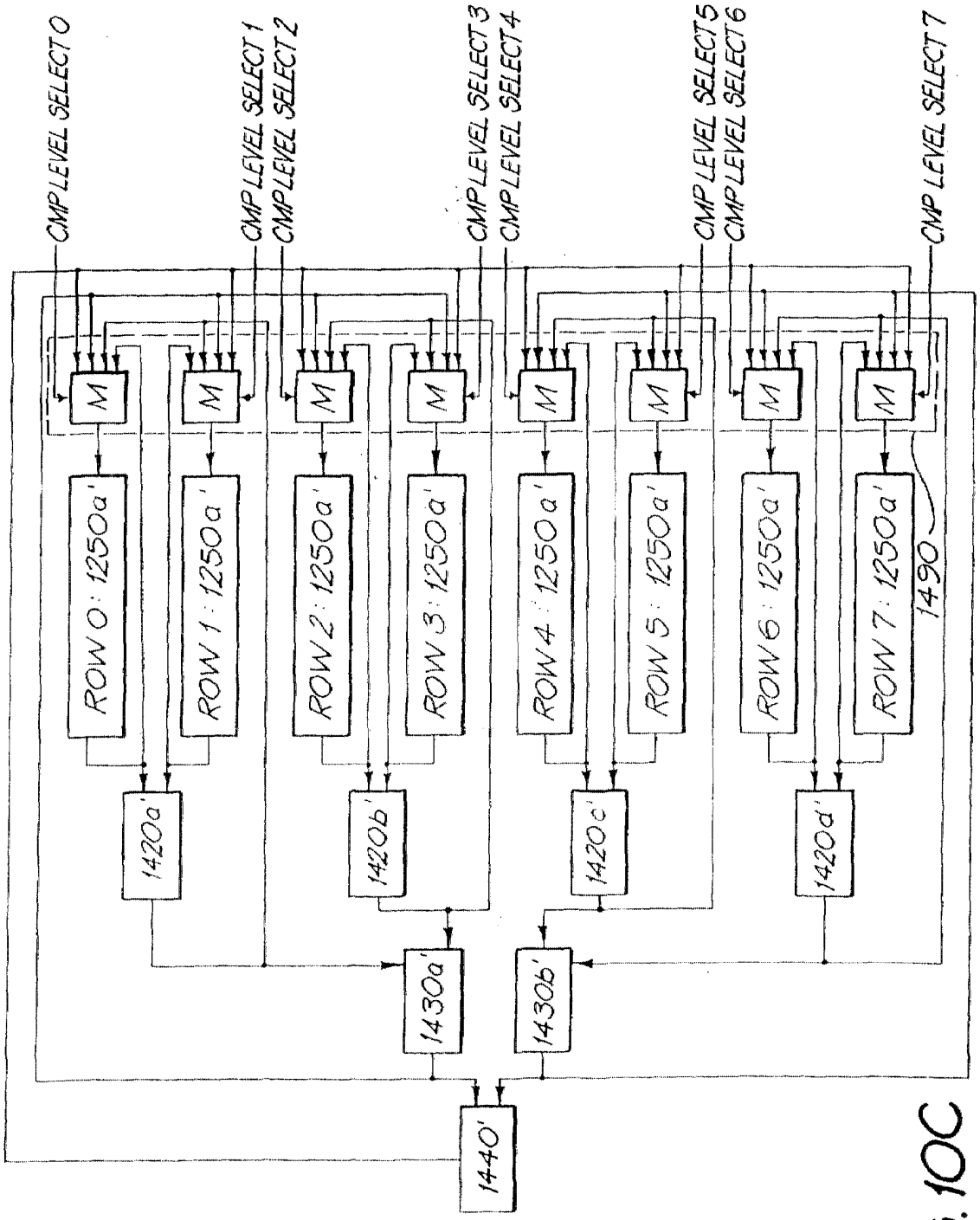

CMP LEVEL SELECT 0
CMP LEVEL SELECT 1
CMP LEVEL SELECT 2
CMP LEVEL SELECT 3
CMP LEVEL SELECT 4
CMP LEVEL SELECT 5
CMP LEVEL SELECT 6
CMP LEVEL SELECT 7
M
ROW 0 : 1250a'
ROW 1 : 1250a'
ROW 2 : 1250a'
ROW 3 : 1250a'
ROW 4 : 1250a'
ROW 5 : 1250a'
ROW 6 : 1250a'
ROW 7 : 1250a'
1490
1420a'
1420b'
1420c'
1420d'
1430a'
1430b'
1440'

FIG. 10C

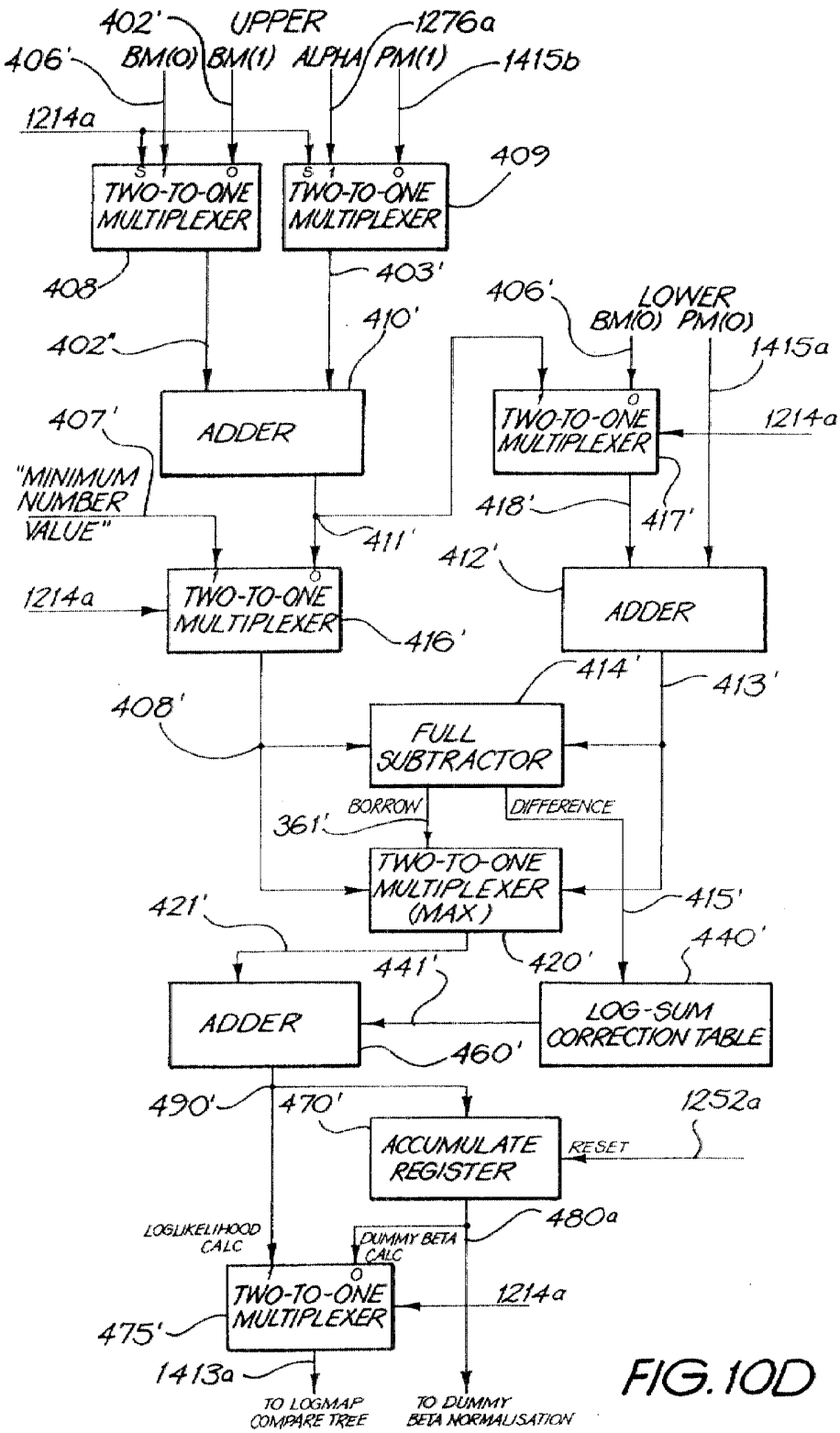

402'
UPPER
1276a
406'
BM(0)
BM(1)
ALPHA
PM(1)
1415b
1214a
409
TWO-TO-ONE MULTIPLEXER
TWO-TO-ONE MULTIPLEXER
408
403'
406'
LOWER
BM(0)
PM(0)
410'
402"
1415a
407'
ADDER
TWO-TO-ONE MULTIPLEXER
1214a
"MINIMUM NUMBER VALUE"
418'
417'
411'
412'
1214a
TWO-TO-ONE MULTIPLEXER
ADDER
416'
414'
413'
408'
FULL SUBTRACTOR
BORROW
DIFFERENCE
361'
TWO-TO-ONE MULTIPLEXER (MAX)
415'
421'
440'
441'
420'
ADDER
LOG-SUM CORRECTION TABLE
460'
490'
470'
1252a
ACCUMULATE REGISTER
RESET
480a
LOGLIKELIHOOD CALC
DUMMY BETA CALC
TWO-TO-ONE MULTIPLEXER
1214a
475'
1413a
TO LOGMAP COMPARE TREE
TO DUMMY BETA NORMALISATION

FIG. 10D

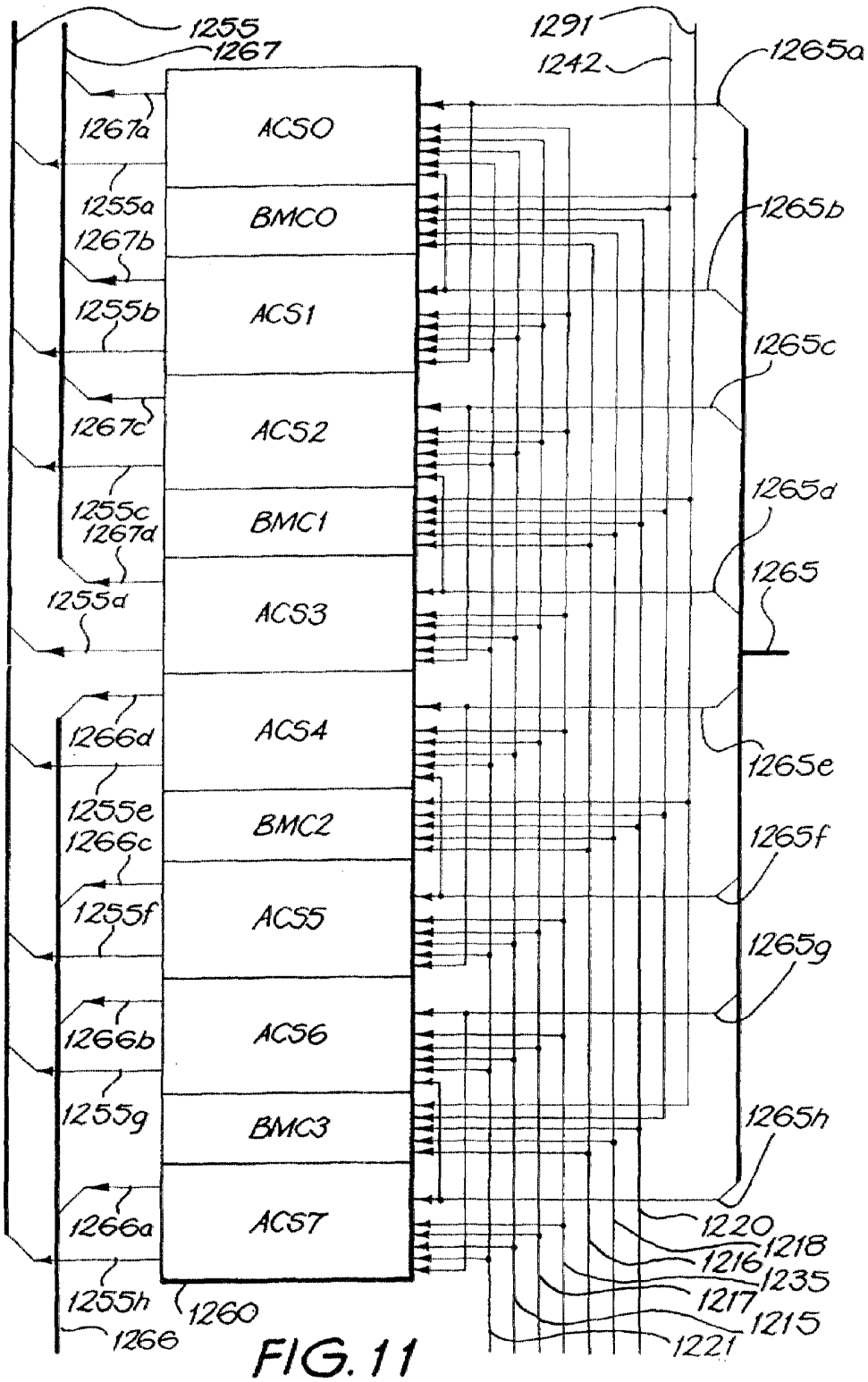

1255
1267
1291
1242
1265a
1267a
ACS0
1255a
BMC0
1265b
1267b
1255b
ACS1
1265c
1267c
ACS2
1265d
1255c
BMC1
1267d
1265
1255d
ACS3
1266d
ACS4
1265e
1255e
BMC2
1265f
1266c
1255f
ACS5
1265g
1266b
ACS6
1255g
BMC3
1265h
1266a
ACS7
1220
1218
1216
1235
1217
1215
1221
1255h
1260
1266

FIG. 11

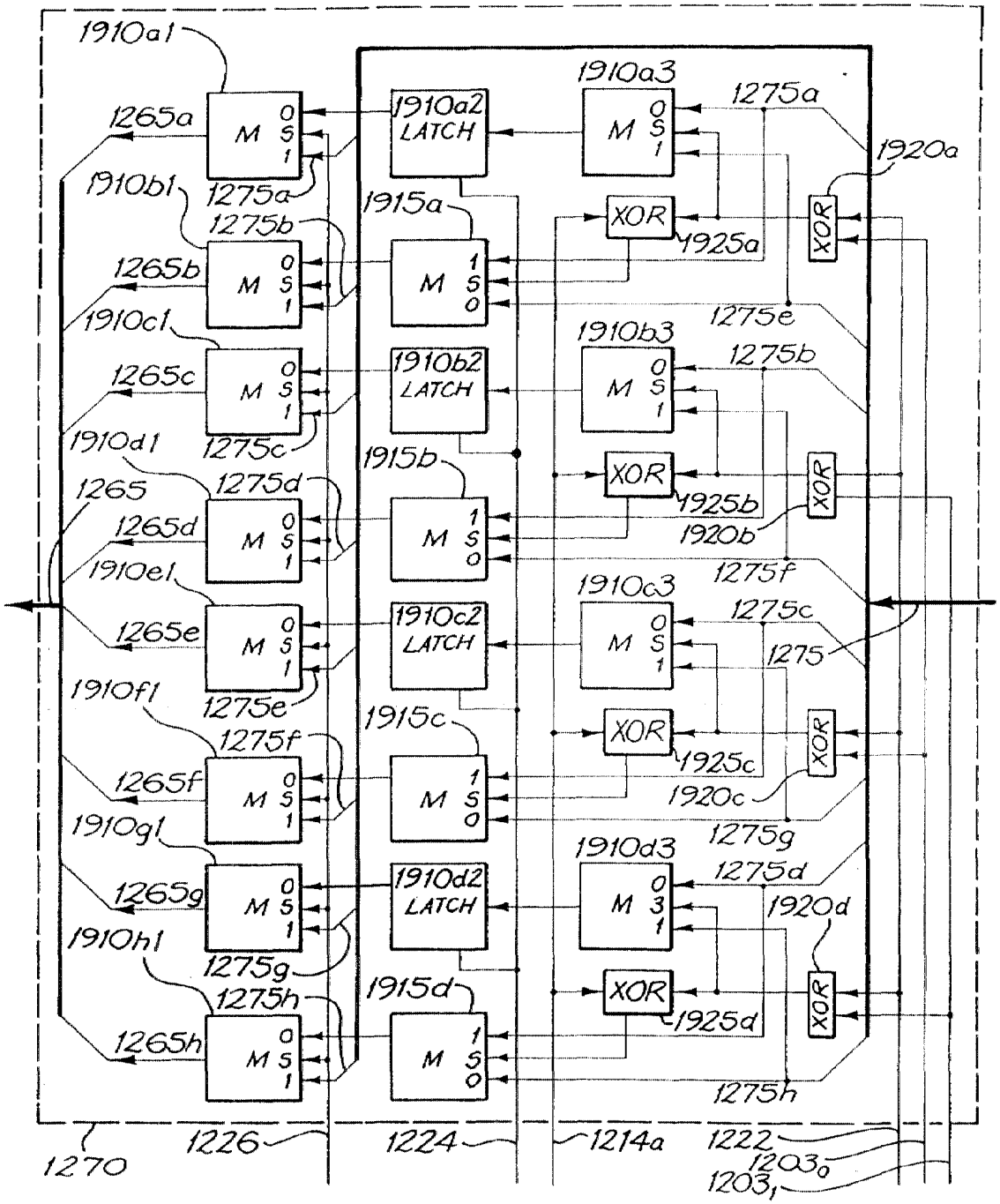

1910a1
1265a
1275a
1910b1
1275b
1265b
1910c1
1265c
1275c
1910d1
1265
1275d
1265d
1910e1
1265e
1275e
1910f1
1275f
1265f
1910g1
1265g
1275g
1910h1
1275h
1265h
1910a2 LATCH
1915a
1910b2 LATCH
1915b
1910c2 LATCH
1915c
1910d2 LATCH
1915d
1910a3
1275a
1920a
XOR
1925a
1275e
1910b3
1275b
XOR
1925b
1920b
1275f
1910c3
1275c
1275
XOR
1925c
1920c
1275g
1910d3
1275d
1920d
XOR
1925d
1275h
1270
1226
1224
1214a
1222
1203₀
1203₁

FIG. 12

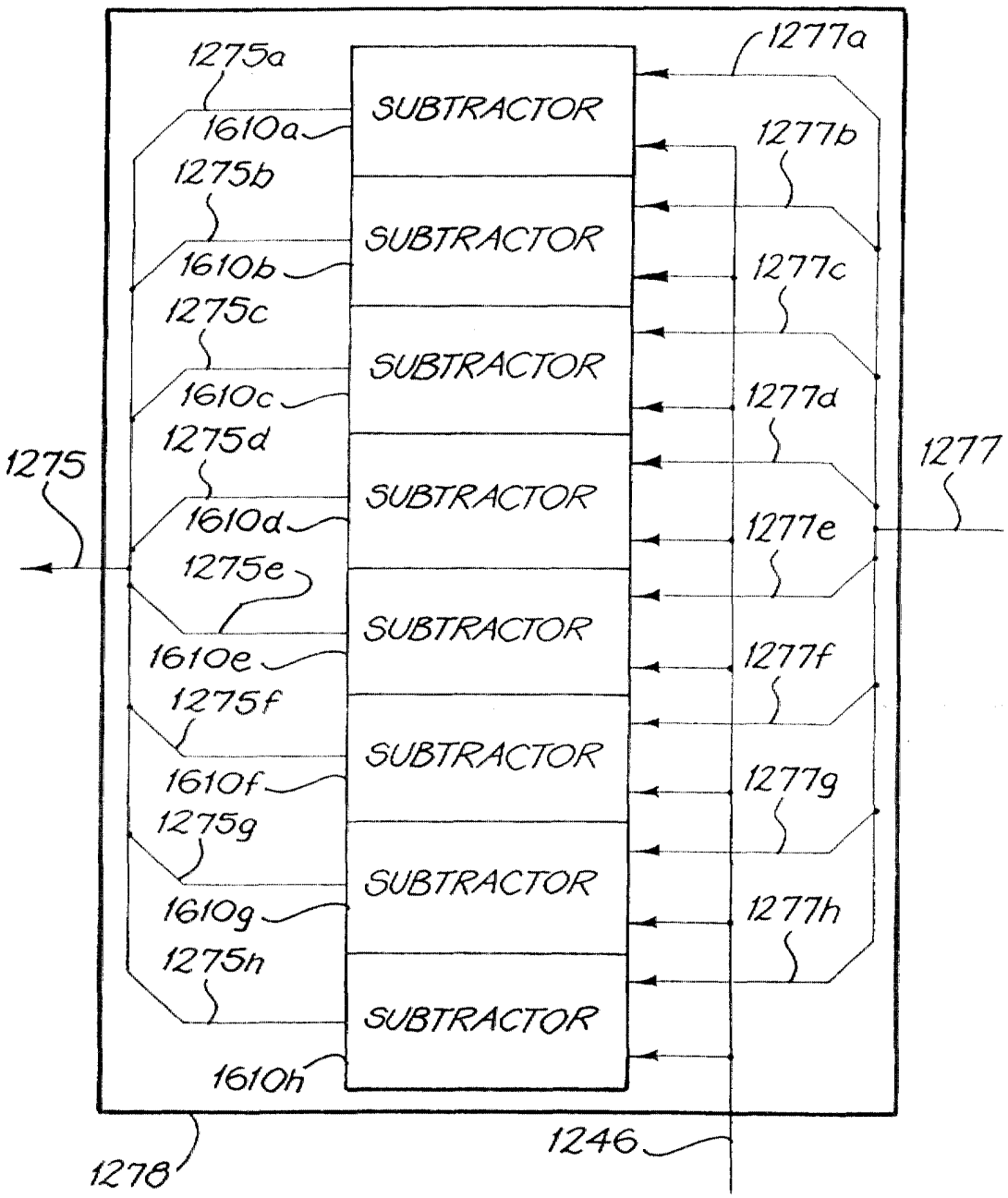

1275a
1610a
SUBTRACTOR
1277a
1277b
1275b
1610b
SUBTRACTOR
1277c
1275c
1610c
SUBTRACTOR
1277d
1275d
1610d
SUBTRACTOR
1277e
1275
1277
1275e
1610e
SUBTRACTOR
1277f
1275f
1610f
SUBTRACTOR
1277g
1275g
1610g
SUBTRACTOR
1277h
1275h
1610h
SUBTRACTOR
1278
1246

FIG. 13

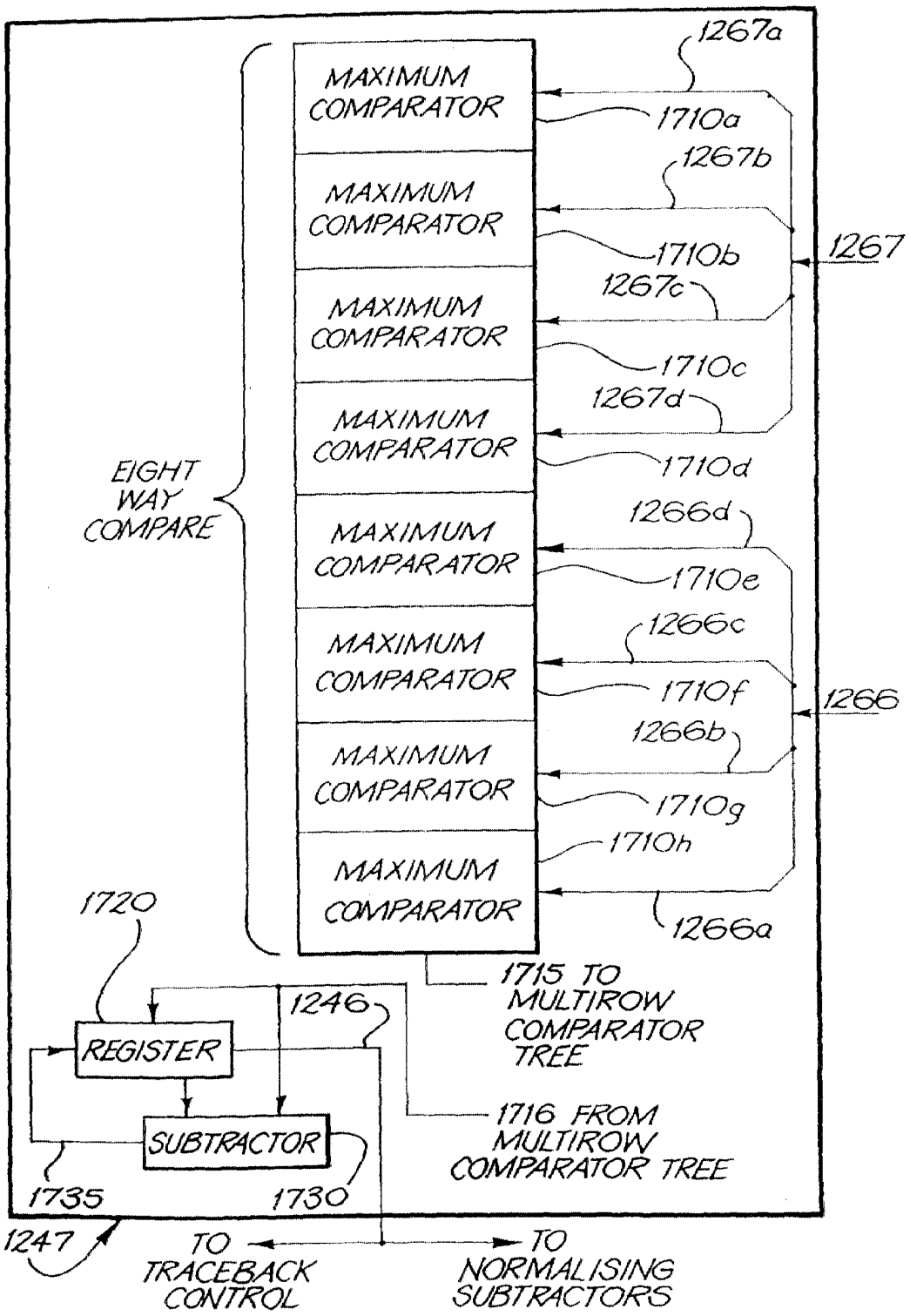

1267a
MAXIMUM COMPARATOR
1710a
1267b
MAXIMUM COMPARATOR
1710b
1267
1267c
MAXIMUM COMPARATOR
1710c
1267d
MAXIMUM COMPARATOR
1710d
EIGHT WAY COMPARE
1266d
MAXIMUM COMPARATOR
1710e
1266c
MAXIMUM COMPARATOR
1710f
1266
1266b
MAXIMUM COMPARATOR
1710g
1710h
MAXIMUM COMPARATOR
1266a
1720
1715 TO MULTIROW COMPARATOR TREE
1246
REGISTER
SUBTRACTOR
1716 FROM MULTIROW COMPARATOR TREE
1735
1730
1247
TO TRACEBACK CONTROL
TO NORMALISING SUBTRACTORS

FIG. 14

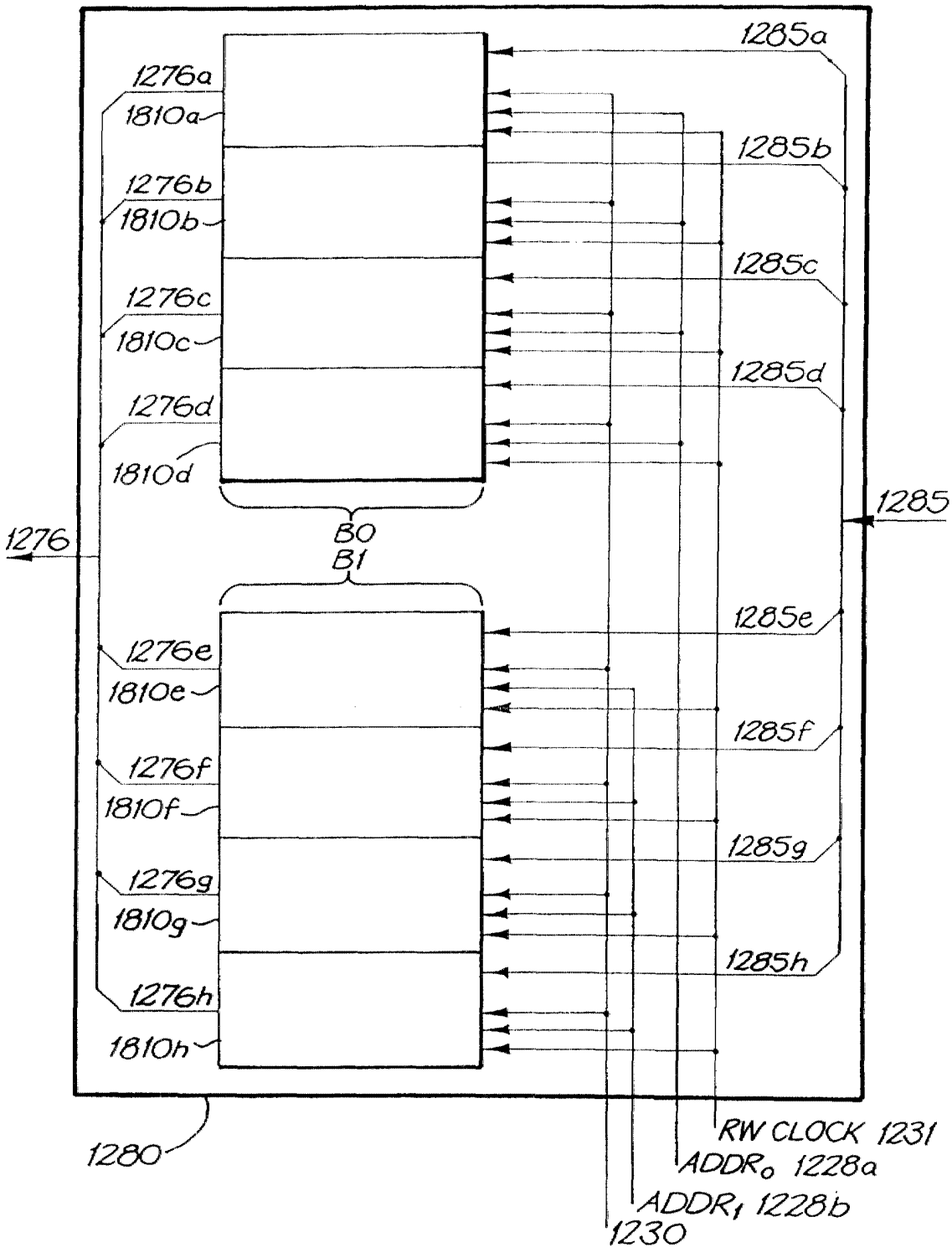

1285a
1276a
1810a
1285b
1276b
1810b
1285c
1276c
1810c
1285d
1276d
1810d
1285
1276
B0
B1
1285e
1276e
1810e
1285f
1276f
1810f
1285g
1276g
1810g
1285h
1276h
1810h
1280
RW CLOCK 1231
ADDR0 1228a
ADDR1 1228b
1230

FIG. 15

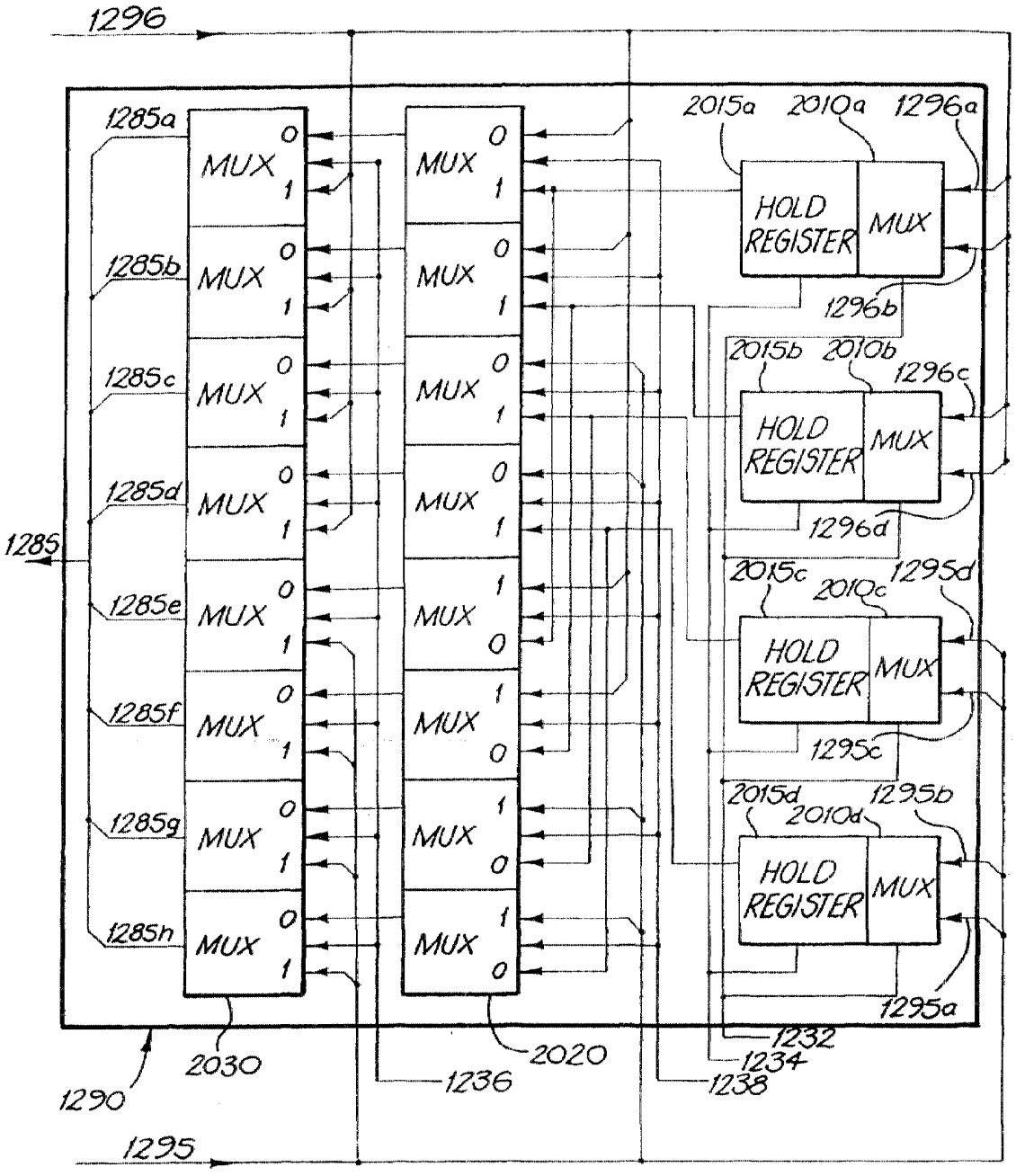
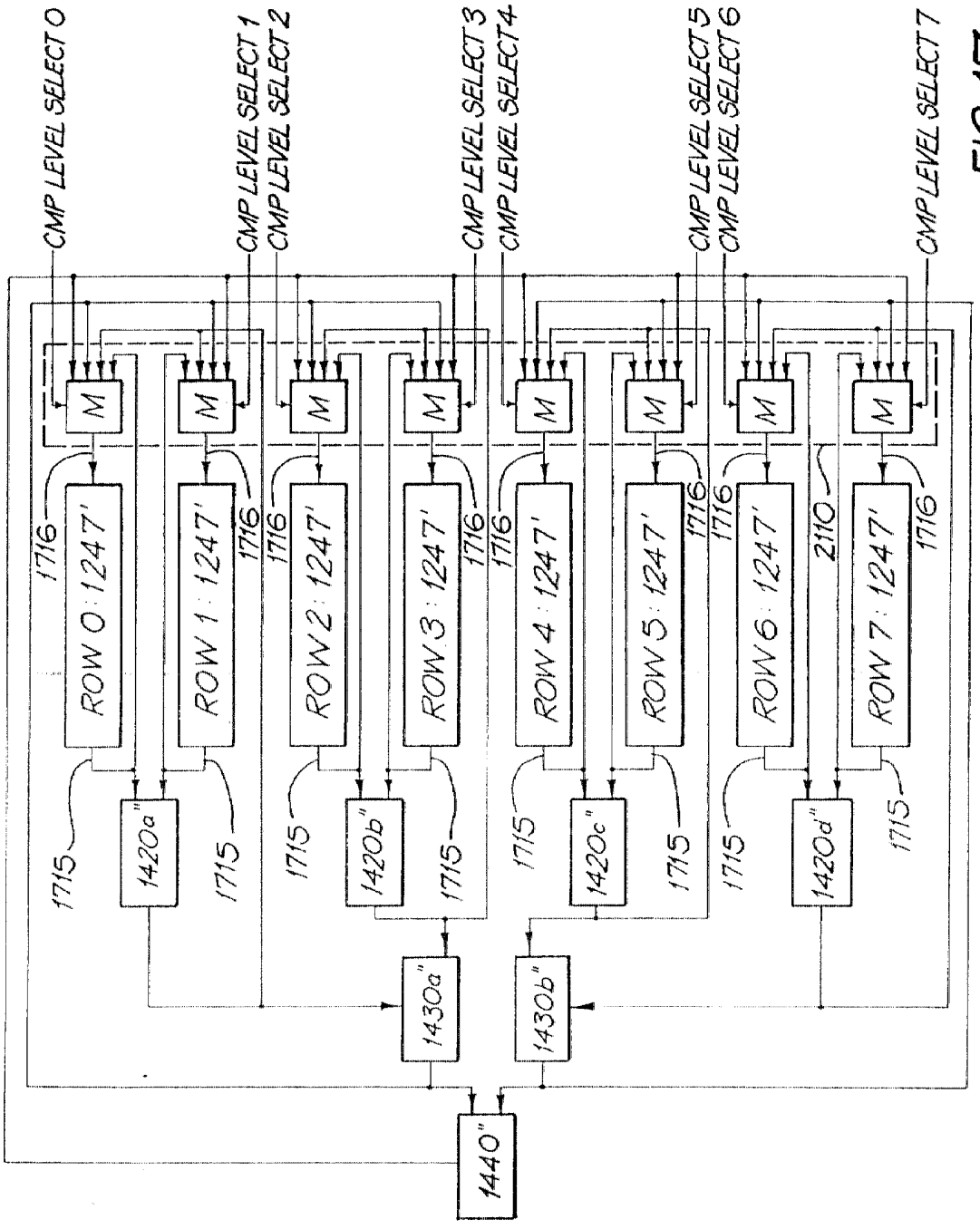

1296
1285a
1285b
1285c
1285d
1285
1285e
1285f
1285g
1285h
MUX
0
1
2015a
2010a
1296a
HOLD REGISTER
1296b
2015b
2010b
1296c
1296d
2015c
2010c
1295d
1295c
2015d
2010d
1295b
1295a
1232
1234
1238
2020
2030
1236
1290
1295

FIG. 16


CMP LEVEL SELECT 0
CMP LEVEL SELECT 1
CMP LEVEL SELECT 2
CMP LEVEL SELECT 3
CMP LEVEL SELECT 4
CMP LEVEL SELECT 5
CMP LEVEL SELECT 6
CMP LEVEL SELECT 7
M
ROW 0 : 1247'
ROW 1 : 1247'
ROW 2 : 1247'
ROW 3 : 1247'
ROW 4 : 1247'
ROW 5 : 1247'
ROW 6 : 1247'
ROW 7 : 1247'
1716
1715
2110
1420a"
1420b"
1420c"
1420d"
1430a"
1430b"
1440"

F/G.17

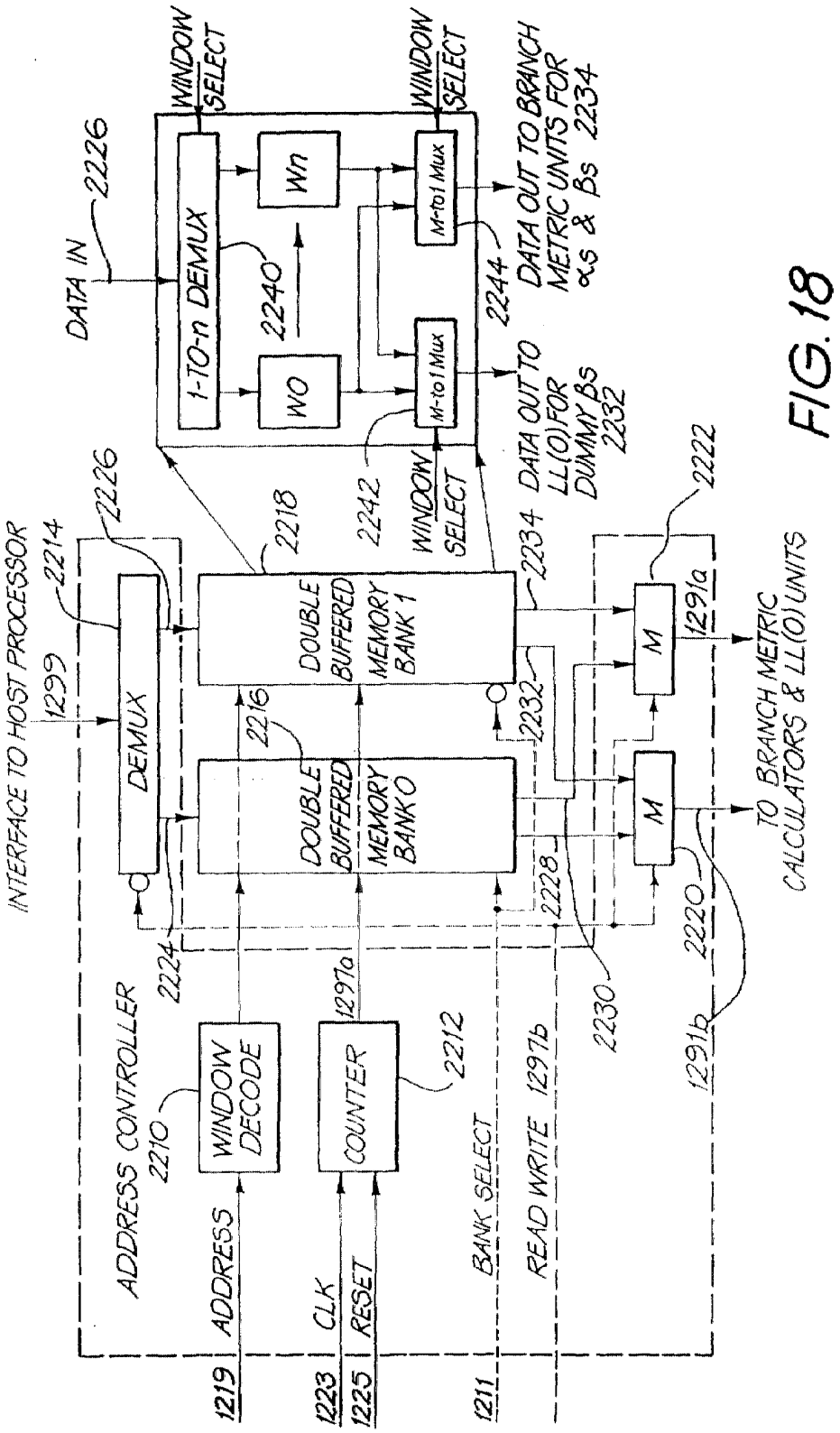

INTERFACE TO HOST PROCESSOR
1299
2214
2226
DEMUX
ADDRESS CONTROLLER
2210
WINDOW DECODE
ADDRESS
1219
CLK
RESET
1223
1225
COUNTER
2212
1297a
1297b
2224
2216
DOUBLE BUFFERED MEMORY BANK 0
DOUBLE BUFFERED MEMORY BANK 1
2218
2228
2230
2232
2234
2242
WINDOW SELECT
BANK SELECT
READ WRITE
1211
DATA IN
2226
1-TO-n DEMUX
WINDOW SELECT
W0
Wn
2240
M-to-1 Mux
M-to-1 Mux
2244
DATA OUT TO BRANCH METRIC UNITS FOR αs & βs 2234
DATA OUT TO LL(0) FOR DUMMY βs 2232
M
M
2222
1291a
1291b
2220
TO BRANCH METRIC CALCULATORS & LL(0) UNITS

FIG. 18

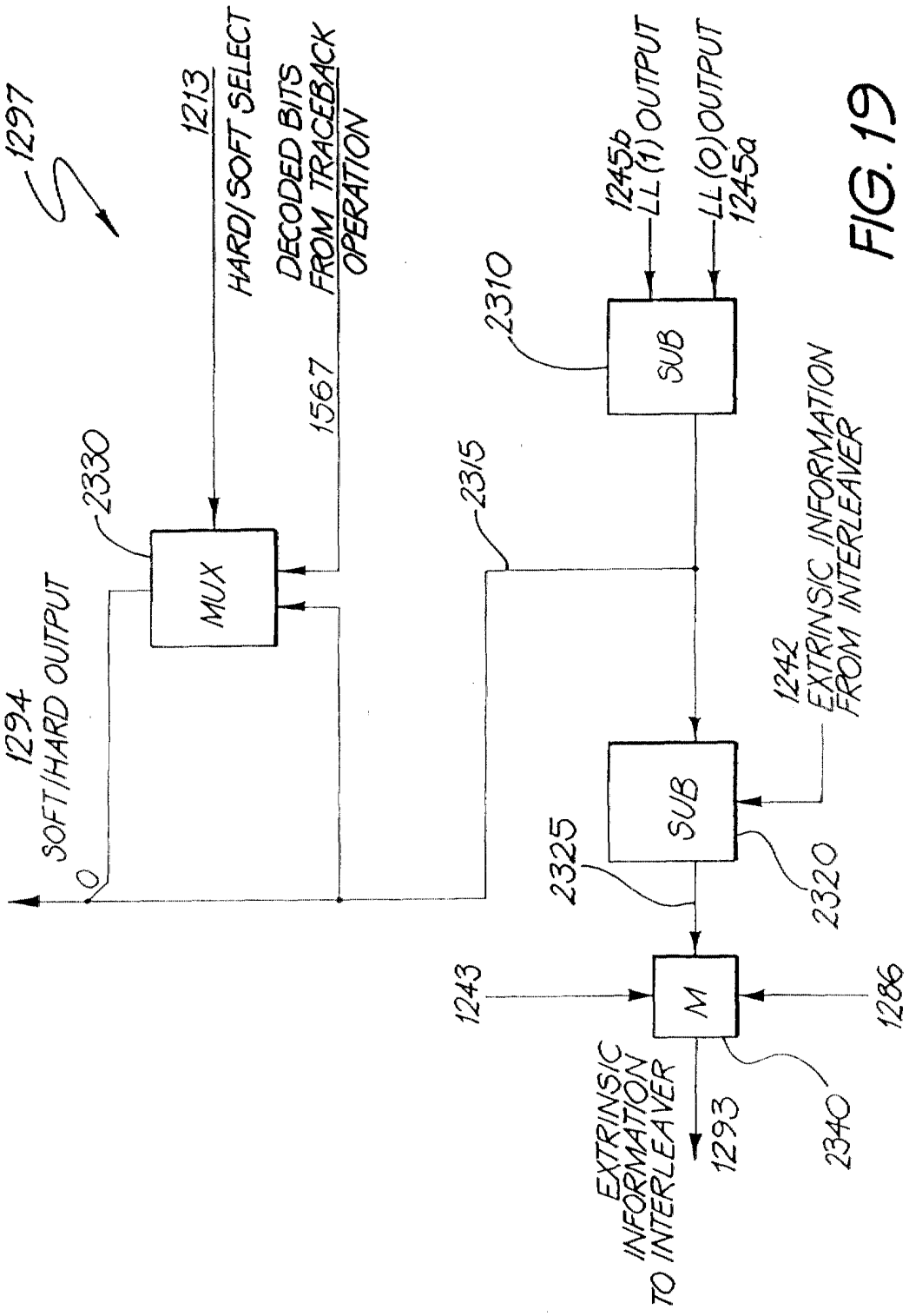

1297
1213
HARD/SOFT SELECT
DECODED BITS FROM TRACEBACK OPERATION
1567
2330
MUX
1294
SOFT/HARD OUTPUT
0
2315
2310
SUB
1245b
LL (1) OUTPUT
LL (0) OUTPUT
1245a
1242
EXTRINSIC INFORMATION FROM INTERLEAVER
SUB
2325
2320
1243
M
1286
EXTRINSIC INFORMATION TO INTERLEAVER
1293
2340

FIG. 19

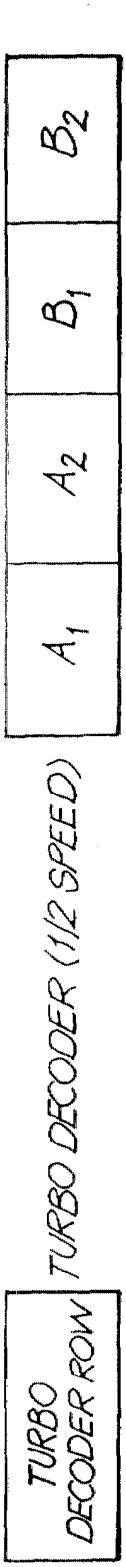


FIG. 20A

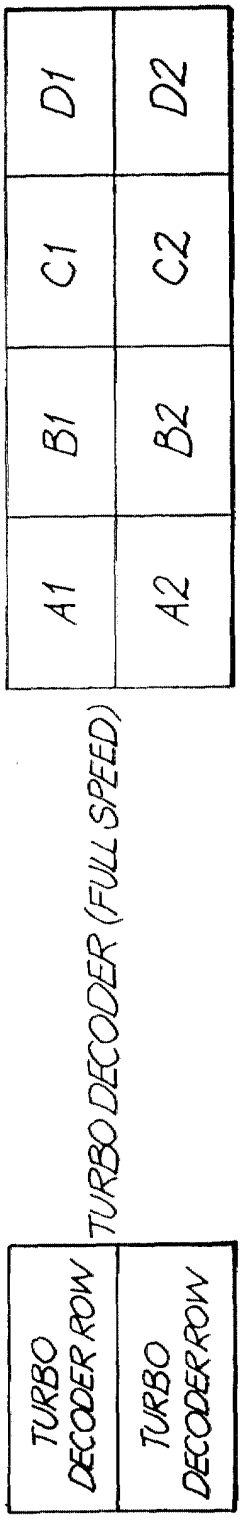


FIG. 20B

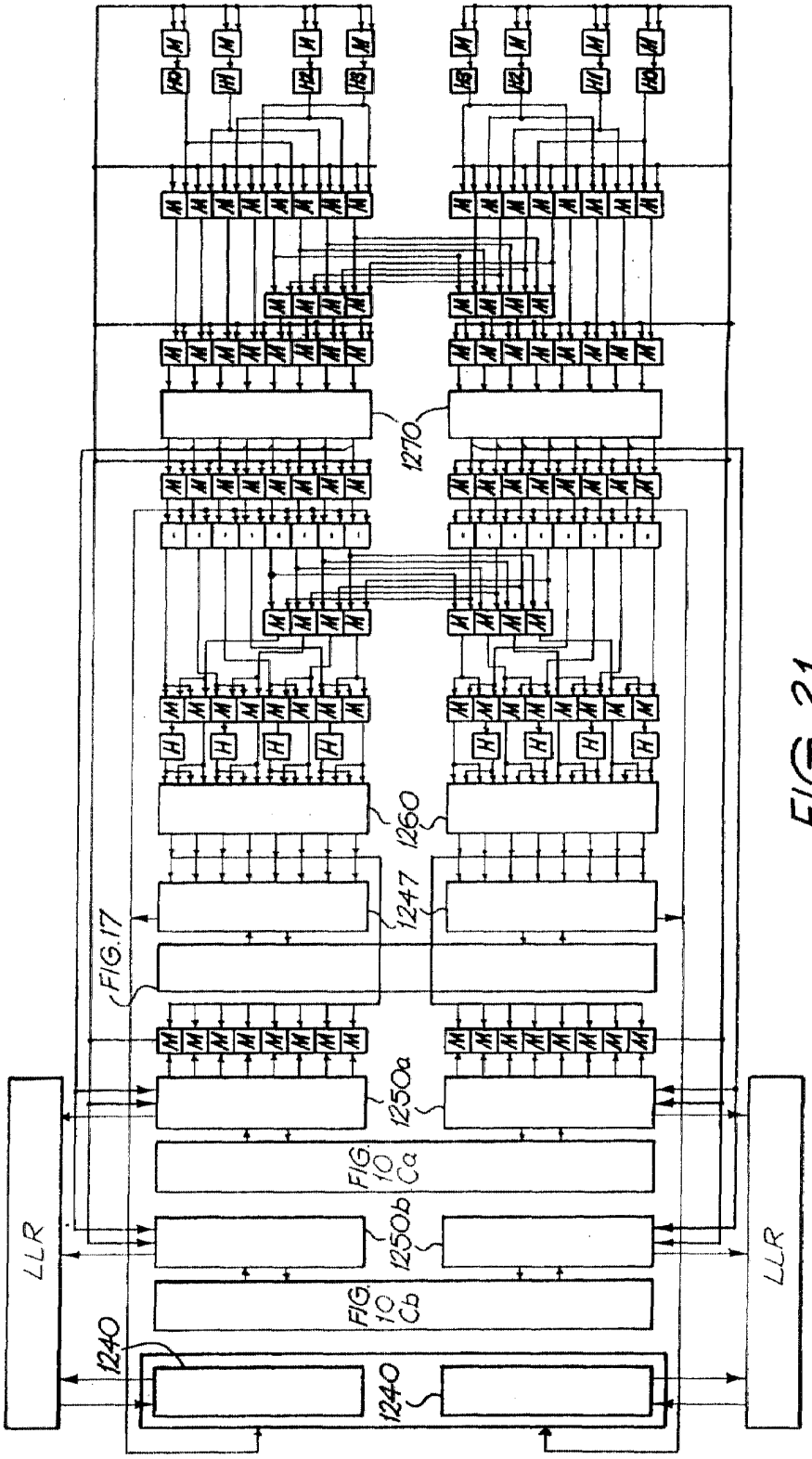

1270
1260
1247
FIG.17
1250a
FIG 10 Ca
1250b
FIG 10 Cb
1240
1240
LLR
LLR

FIG.21

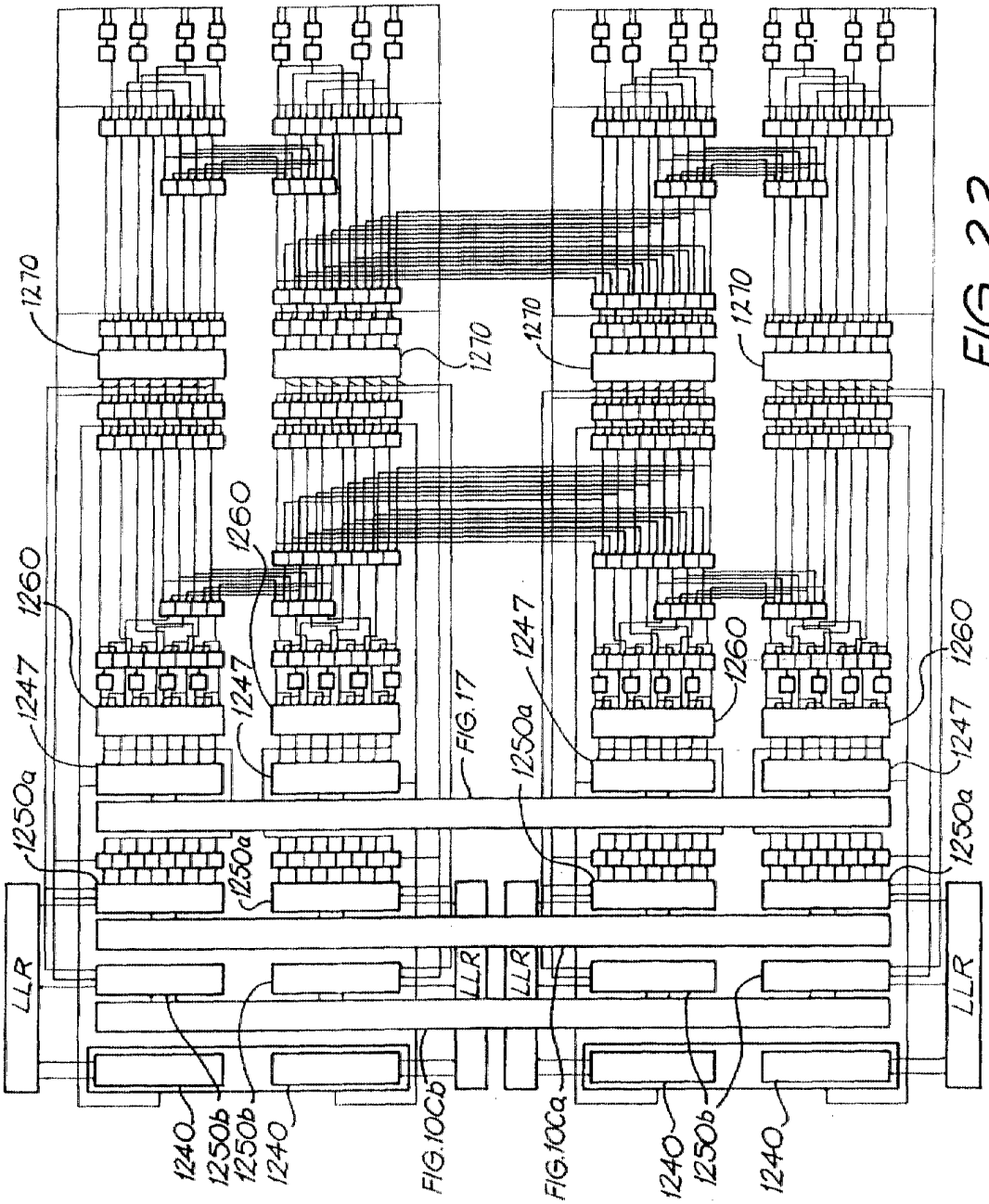

1270
1270
1270
1270
1260
1260
1260
1260
1247
1247
1247
1247
1250a
1250a
1250a
1250a
LLR
LLR
LLR
LLR
1240
1250b
1250b
1240
1240
1250b
1240
FIG.17
FIG.10Cb
FIG.10Ca

FIG. 22

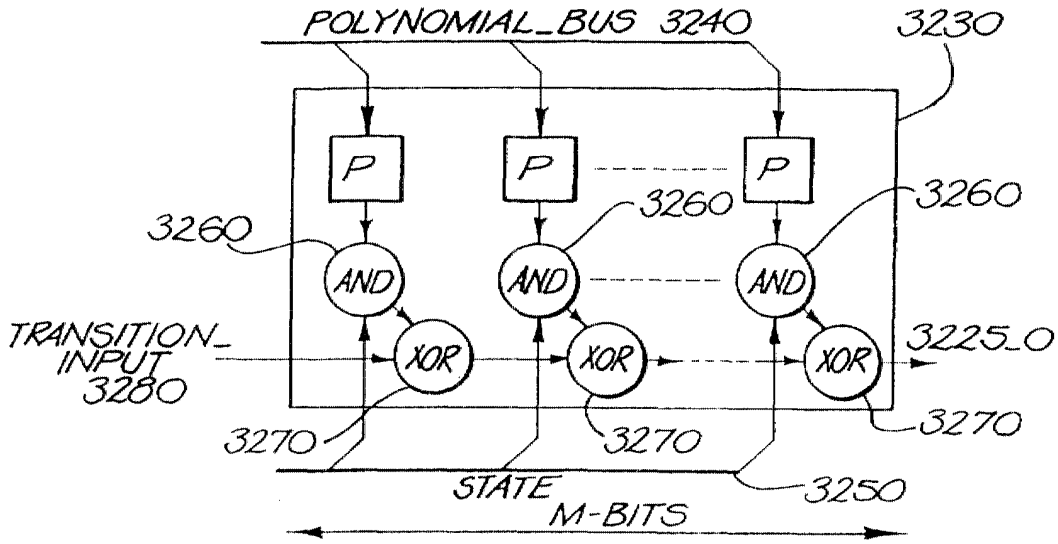


FIG. 23A

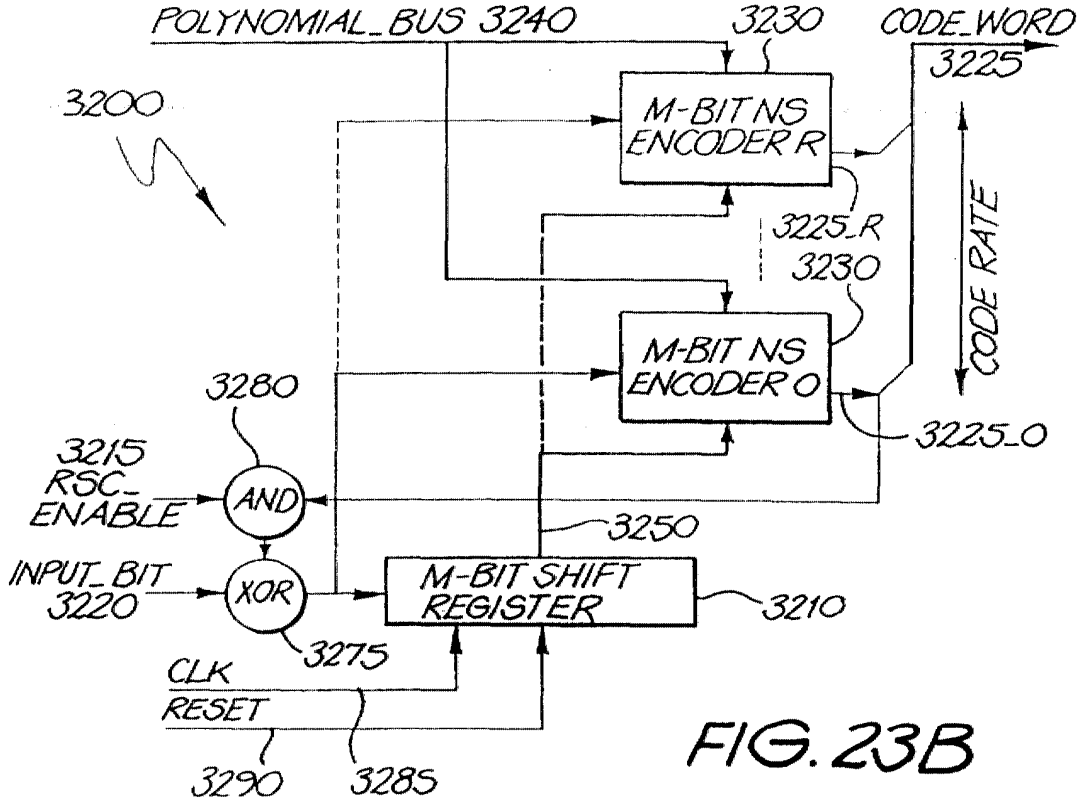


FIG. 23B

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 01 30 7697

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
X	WO 99 52216 A (SAMSUNG ELECTRONICS CO LTD) 14 October 1999 (1999-10-14) * the whole document * ---	1,10	H03M13/27
A	EP 0 963 048 A (UNITED KINGDOM GOVERNMENT) 8 December 1999 (1999-12-08) * the whole document * ---	1-10	
A	WO 00 38366 A (ERICSSON TELEFON AB L M) 29 June 2000 (2000-06-29) * the whole document * ---	1-10	
A	US 5 327 440 A (RAE JAMES W ET AL) 5 July 1994 (1994-07-05) * the whole document * ---	1-10	
A	PIETROBON S S: "IMPLEMENTATION AND PERFORMANCE OF A TURBO/MAP DECODER" INTERNATIONAL JOURNAL OF SATELLITE COMMUNICATIONS, JOHN WILEY AND SONS, US, vol. 16, no. 1, 1998, pages 23-46, XP000856961 ISSN: 0737-2884 * the whole document * ---	1-10	TECHNICAL FIELDS SEARCHED (Int.Cl.7) H03M
A	US 5 933 462 A (SINDHUSHAYANA NAGABHUSHANA T ET AL) 3 August 1999 (1999-08-03) * the whole document * ---	1-10	
A	US 6 115 436 A (KELLAM GEORGE ET AL) 5 September 2000 (2000-09-05) * the whole document * ---	1-10	
A	US 5 068 859 A (DOLINAR JR SAMUEL J ET AL) 26 November 1991 (1991-11-26) * the whole document * ---	1-10	
	-/--		

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	12 March 2002	Barel-Faucheux, C

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 01 30 7697

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
P,A	WO 01 26257 A (SAMSUNG ELECTRONICS CO LTD) 12 April 2001 (2001-04-12) * the whole document * ---	1-10	
P,A	GB 2 357 938 A (NOKIA NETWORKS OY) 4 July 2001 (2001-07-04) * the whole document * -----	1-10	

TECHNICAL FIELDS SEARCHED (Int.Cl.7)

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	12 March 2002	Barel-Faucheux, C

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 01 30 7697

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

12-03-2002

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9952216 A	14-10-1999	AU 735133 B2	28-06-2001
		AU 3057499 A	25-10-1999
		BR 9906303 A	05-12-2000
		CN 1301431 T	27-06-2001
		EP 0998789 A2	10-05-2000
		JP 2000515715 T	21-11-2000
		WO 9952216 A2	14-10-1999
		US 6166667 A	26-12-2000
EP 0963048 A	08-12-1999	EP 0963049 A2	08-12-1999
		EP 0963048 A2	08-12-1999
		US 6339834 B1	15-01-2002
WO 0038366 A	29-06-2000	AU 1978000 A	12-07-2000
		WO 0038366 A1	29-06-2000
		EP 1142183 A1	10-10-2001
		US 6343368 B1	29-01-2002
US 5327440 A	05-07-1994	JP 2718859 B2	25-02-1998
		JP 5210921 A	20-08-1993
US 5933462 A	03-08-1999	AU 722477 B2	03-08-2000
		AU 5104598 A	29-05-1998
		CN 1236507 A	24-11-1999
		EP 0937336 A1	25-08-1999
		JP 2001503588 T	13-03-2001
		TW 431097 B	21-04-2001
		WO 9820617 A1	14-05-1998
		ZA 9709958 A	25-05-1998
US 6115436 A	05-09-2000	AU 1942799 A	19-07-1999
		CN 1283331 T	07-02-2001
		EP 1044507 A1	18-10-2000
		JP 2002500463 T	08-01-2002
		WO 9934520 A1	08-07-1999
US 5068859 A	26-11-1991	NONE	
WO 0126257 A	12-04-2001	AU 7690900 A	10-05-2001
		BR 0007197 A	04-09-2001
		CN 1327653 T	19-12-2001
		EP 1135877 A1	26-09-2001
		WO 0126257 A1	12-04-2001
GB 2357938 A	04-07-2001	AU 2011201 A	09-07-2001
		WO 0148966 A1	05-07-2001

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82